

new/usr/src/cmd/lofiadm/main.c

1

```
*****
54666 Sun Jan 17 00:44:07 2016
new/usr/src/cmd/lofiadm/main.c
5857 add -o option to lofiadm
*****

1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */
21 /*
22 * Copyright 2009 Sun Microsystems, Inc. All rights reserved.
23 * Use is subject to license terms.
24 * Copyright 2012 Joyent, Inc. All rights reserved.
25 *
26 * Copyright 2013 Nexenta Systems, Inc. All rights reserved.
27 * Copyright (c) 2014 Gary Mills
28 * Copyright (c) 2016 Andrey Sokolov
29 #endif /* ! codereview */
30 */
31
32 /*
33 * lofiadm - administer lofi(7d). Very simple, add and remove file<->device
34 * associations, and display status. All the ioctl's are private between
35 * lofi and lofiadm, and so are very simple - device information is
36 * communicated via a minor number.
37 */
38
39 #include <sys/types.h>
40 #include <sys/param.h>
41 #include <sys/lofi.h>
42 #include <sys/stat.h>
43 #include <sys/sysmacros.h>
44 #include <netinet/in.h>
45 #include <stdio.h>
46 #include <fcntl.h>
47 #include <locale.h>
48 #include <string.h>
49 #include <strings.h>
50 #include <errno.h>
51 #include <stdlib.h>
52 #include <unistd.h>
53 #include <stropts.h>
54 #include <libdevinfo.h>
55 #include <libgen.h>
56 #include <ctype.h>
57 #include <dlfcn.h>
58 #include <limits.h>
59 #include <security/cryptoki.h>
60 #include <cryptoutil.h>
61 #include <sys/crypto/ioctl.h>
```

new/usr/src/cmd/lofiadm/main.c

2

```
62 #include <sys/crypto/ioctladmin.h>
63 #include "utils.h"
64 #include <LzmaEnc.h>
65
66 /* Only need the IV len #defines out of these files, nothing else. */
67 #include <aes/aes_impl.h>
68 #include <des/des_impl.h>
69 #include <blowfish/blowfish_impl.h>
70
71 static const char USAGE[] =
72 "Usage: %s [-r] -a file [ device ]\n"
73 "        %s [-r] [-o] -c crypto_algorithm -a file [device]\n"
74 "        %s [-r] -c crypto_algorithm -a file [device]\n"
75 "        %s [-r] -c crypto_algorithm -k raw_key_file -a file [device]\n"
76 "        %s [-r] -c crypto_algorithm -T [token]:[manuf]:[serial]:key "
77 "        %s [-r] -c crypto_algorithm -T [token]:[manuf]:[serial]:key "
78 "        %s [-r] -c crypto_algorithm -k wrapped_key_file -a file [device]\n"
79 "        %s [-r] -c crypto_algorithm -e -a file [device]\n"
80 "        %s -d file | device\n"
81 "        %s -C [gzip|gzip-6|gzip-9|lzma] [-s segment_size] file\n"
82 "        %s -U file\n"
83 "        %s [ file | device ]\n";
84
85 typedef struct token_spec {
86     char    *name;
87     char    *mfr;
88     char    *serno;
89     char    *key;
90 } token_spec_t;
91
92 #ifndef unchanged_portion_omitted
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2
```

```

869         goto cleanup;

871     /* get user passphrase with 8 byte minimum */
872     if (pkcs11_get_pass(NULL, &pass, &passlen, MIN_PASSLEN,
873         with_confirmation) < 0) {
874         if (pkcs11_get_pass(NULL, &pass, &passlen, MIN_PASSLEN, B_TRUE) < 0) {
875             die(gettext("passphrases do not match\n"));
876         }
877     }

878     /*
879     * salt should not be NULL, or else pkcs11_PasswdToKey() will
880     * complain about CKR_MECHANISM_PARAM_INVALID; the following is
881     * to make up for not having a salt until a proper one is used
882     */
883     salt = pass;
884     saltlen = passlen;

885     klen = cipher->max_keysize;
886     rv = pkcs11_PasswdToKey(sess, pass, passlen, salt, saltlen, ktype,
887         cipher->max_keysize, &kvalue, &klen);

889     (void) C_CloseSession(sess);

891     if (rv != CKR_OK) {
892         goto cleanup;
893     }

895     /* assert(klen == cipher->max_keysize); */
896     *raw_key_sz = klen;
897     *raw_key = (char *)kvalue;
898     return;

900 cleanup:
901     die(gettext("failed to generate %s key from passphrase: %s"),
902         cipher->alias, pkcs11_strerror(rv));
903 }

```

unchanged_portion_omitted

```

1802 int
1803 main(int argc, char *argv[])
1804 {
1805     int    lfd;
1806     int    c;
1807     const char *devicename = NULL;
1808     const char *filename = NULL;
1809     const char *alname = COMPRESS_ALGORITHM;
1810     int    openflag;
1811     int    minor;
1812     int    compress_index;
1813     uint32_t segsize = SEGSIZE;
1814     static char *lofictl = "/dev/" LOFI_CTL_NAME;
1815     boolean_t force = B_FALSE;
1816     const char *pname;
1817     boolean_t errflag = B_FALSE;
1818     boolean_t addflag = B_FALSE;
1819     boolean_t rdflag = B_FALSE;
1820     boolean_t deleteflag = B_FALSE;
1821     boolean_t ephflag = B_FALSE;
1822     boolean_t compressflag = B_FALSE;
1823     boolean_t uncompressflag = B_FALSE;
1824     /* the next two work together for -c, -k, -T, -e options only */
1825     boolean_t need_crypto = B_FALSE; /* if any -c, -k, -T, -e */
1826     boolean_t cipher_only = B_TRUE; /* if -c only */
1827     boolean_t with_confirmation = B_TRUE;
1828 #endif /* ! codereview */
1829     const char *keyfile = NULL;

```

```

1830     mech_alias_t *cipher = NULL;
1831     token_spec_t *token = NULL;
1832     char *rkey = NULL;
1833     size_t rksz = 0;
1834     char realfilename[MAXPATHLEN];

1836     pname = getpname(argv[0]);

1838     (void) setlocale(LC_ALL, "");
1839     (void) textdomain(TEXT_DOMAIN);

1841     while ((c = getopt(argc, argv, "a:c:Cd:efk:ors:T:U")) != EOF) {
1842         while ((c = getopt(argc, argv, "a:c:Cd:efk:ors:T:U")) != EOF) {
1843             switch (c) {
1844                 case 'a':
1845                     addflag = B_TRUE;
1846                     if ((filename = realpath(optarg, realfilename)) == NULL)
1847                         die("%s", optarg);
1848                     if (((argc - optind) > 0) && (*argv[optind] != '-')) {
1849                         /* optional device */
1850                         devicename = argv[optind];
1851                         optind++;
1852                     }
1853                     break;
1854                 case 'c':
1855                     compressflag = B_TRUE;
1856                     if (((argc - optind) > 1) && (*argv[optind] != '-')) {
1857                         /* optional algorithm */
1858                         alname = argv[optind];
1859                         optind++;
1860                     }
1861                     check_algorithm_validity(alname, &compress_index);
1862                     break;
1863                 case 'C':
1864                     /* is the chosen cipher allowed? */
1865                     if ((cipher = ciph2mech(optarg)) == NULL) {
1866                         errflag = B_TRUE;
1867                         warn(gettext("cipher %s not allowed\n"),
1868                             optarg);
1869                     }
1870                     need_crypto = B_TRUE;
1871                     /* cipher_only is already set */
1872                     break;
1873                 case 'd':
1874                     deleteflag = B_TRUE;
1875                     minor = name_to_minor(optarg);
1876                     if (minor != 0)
1877                         devicename = optarg;
1878                     else {
1879                         if ((filename = realpath(optarg,
1880                             realfilename)) == NULL)
1881                             die("%s", optarg);
1882                     }
1883                     break;
1884                 case 'e':
1885                     ephflag = B_TRUE;
1886                     need_crypto = B_TRUE;
1887                     cipher_only = B_FALSE; /* need to unset cipher_only */
1888                     break;
1889                 case 'f':
1890                     force = B_TRUE;
1891                     break;
1892                 case 'k':
1893                     keyfile = optarg;
1894                     need_crypto = B_TRUE;
1895                     cipher_only = B_FALSE; /* need to unset cipher_only */

```

```

1895         break;
1896     case 'r':
1897         rdflag = B_TRUE;
1898         break;
1899     case 's':
1900         segsize = convert_to_num(optarg);
1901         if (segsize < DEV_BSIZE || !ISP2(segsize))
1902             die(gettext("segment size %s is invalid "
1903                 "or not a multiple of minimum block "
1904                 "size %ld\n"), optarg, DEV_BSIZE);
1905         break;
1906     case 'T':
1907         if ((token = parsetoken(optarg)) == NULL) {
1908             errflag = B_TRUE;
1909             warn(
1910                 gettext("invalid token key specifier %s\n"),
1911                 optarg);
1912         }
1913         need_crypto = B_TRUE;
1914         cipher_only = B_FALSE; /* need to unset cipher_only */
1915         break;
1916     case 'U':
1917         uncompressflag = B_TRUE;
1918         break;
1919     case 'o':
1920         with_confirmation = B_FALSE;
1921         break;
1922 #endif /* ! codereview */
1923     case '?':
1924     default:
1925         errflag = B_TRUE;
1926         break;
1927 }
1928
1930 /* Check for mutually exclusive combinations of options */
1931 if (errflag ||
1932     (addflag && deleteflag) ||
1933     (rdflag && !addflag) ||
1934     (!addflag && need_crypto) ||
1935     (!with_confirmation && (!cipher_only || !need_crypto)) ||
1936 #endif /* ! codereview */
1937     ((compressflag || uncompressflag) && (addflag || deleteflag)))
1938     usage(pname);
1939
1940 /* ephemeral key, and key from either file or token are incompatible */
1941 if (ephflag && (keyfile != NULL || token != NULL)) {
1942     die(gettext("ephemeral key cannot be used with keyfile "
1943         "or token key\n"));
1944 }
1945
1946 /*
1947 * "-c" but no "-k", "-T", "-e", or "-T -k" means derive key from
1948 * command line passphrase
1949 */
1950
1951 switch (argc - optind) {
1952 case 0: /* no more args */
1953     if (compressflag || uncompressflag) /* needs filename */
1954         usage(pname);
1955     break;
1956 case 1:
1957     if (addflag || deleteflag)
1958         usage(pname);
1959     /* one arg means compress/uncompress the file ... */
1960     if (compressflag || uncompressflag) {

```

```

1961         if ((filename = realpath(argv[optind],
1962             realfilename)) == NULL)
1963             die("%s", argv[optind]);
1964         /* ... or without options means print the association */
1965     } else {
1966         minor = name_to_minor(argv[optind]);
1967         if (minor != 0)
1968             devicename = argv[optind];
1969         else {
1970             if ((filename = realpath(argv[optind],
1971                 realfilename)) == NULL)
1972                 die("%s", argv[optind]);
1973         }
1974     }
1975     break;
1976 default:
1977     usage(pname);
1978     break;
1979 }
1980
1981 if (addflag || compressflag || uncompressflag)
1982     check_file_validity(filename);
1983
1984 if (filename && !valid_abspath(filename))
1985     exit(E_ERROR);
1986
1987 /*
1988 * Here, we know the arguments are correct, the filename is an
1989 * absolute path, it exists and is a regular file. We don't yet
1990 * know that the device name is ok or not.
1991 */
1992
1993 openflag = O_EXCL;
1994 if (addflag || deleteflag || compressflag || uncompressflag)
1995     openflag |= O_RDWR;
1996 else
1997     openflag |= O_RDONLY;
1998 lfd = open(lofictl, openflag);
1999 if (lfd == -1) {
2000     if ((errno == EPERM) || (errno == EACCES)) {
2001         die(gettext("you do not have permission to perform "
2002             "that operation.\n"));
2003     } else {
2004         die(gettext("open: %s"), lofictl);
2005     }
2006     /*NOTREACHED*/
2007 }
2008
2009 /*
2010 * No passphrase is needed for ephemeral key, or when key is
2011 * in a file and not wrapped by another key from a token.
2012 * However, a passphrase is needed in these cases:
2013 * 1. cipher with no ephemeral key, key file, or token,
2014 *    in which case the passphrase is used to build the key
2015 * 2. token with an optional cipher or optional key file,
2016 *    in which case the passphrase unlocks the token
2017 * If only the cipher is specified, reconfirm the passphrase
2018 * to ensure the user hasn't mis-entered it. Otherwise, the
2019 * token will enforce the token passphrase.
2020 */
2021 if (need_crypto) {
2022     CK_SESSION_HANDLE sess;
2023
2024     /* pick a cipher if none specified */
2025     if (cipher == NULL)
2026         cipher = DEFAULT_CIPHER;

```

```
2028         if (!kernel_cipher_check(cipher))
2029             die(gettext(
2030                 "use \"cryptoadm list -m\" to find available \"
2031                 \"mechanisms\\n\"));
2033         init_crypto(token, cipher, &sess);
2035         if (cipher_only) {
2036             getkeyfromuser(cipher, &rkey, &rksz, with_confirmation);
1858             getkeyfromuser(cipher, &rkey, &rksz);
2037         } else if (token != NULL) {
2038             getkeyfromtoken(sess, token, keyfile, cipher,
2039                             &rkey, &rksz);
2040         } else {
2041             /* this also handles ephemeral keys */
2042             getkeyfromfile(keyfile, cipher, &rkey, &rksz);
2043         }
2045         end_crypto(sess);
2046     }
2048     /*
2049     * Now to the real work.
2050     */
2051     if (addflag)
2052         add_mapping(lfd, devicename, filename, cipher, rkey, rksz,
2053                     rdflag);
2054     else if (compressflag)
2055         lofi_compress(&lfd, filename, compress_index, segsize);
2056     else if (uncompressflag)
2057         lofi_uncompress(lfd, filename);
2058     else if (deleteflag)
2059         delete_mapping(lfd, devicename, filename, force);
2060     else if (filename || devicename)
2061         print_one_mapping(lfd, devicename, filename);
2062     else
2063         print_mappings(lfd);
2065     if (lfd != -1)
2066         (void) close(lfd);
2067     closelib();
2068     return (E_SUCCESS);
2069 }
_____unchanged_portion_omitted_
```

new/usr/src/man/man1m/lofiadm.1m

1

18522 Sun Jan 17 00:44:08 2016

new/usr/src/man/man1m/lofiadm.1m

5857 add -o option to lofiadm

```
1 \" te
2.\" Copyright (c) 2016 Andrey Sokolov
3 #endif /* ! codereview */
4.\" Copyright 2013 Nexenta Systems, Inc. All rights reserved.
5.\" Copyright (c) 2008, Sun Microsystems, Inc. All Rights Reserved
6.\" The contents of this file are subject to the terms of the Common Development
7.\" See the License for the specific language governing permissions and limitat
8.\" the fields enclosed by brackets \"[]\" replaced with your own identifying info
9.TH LOFIADM 1M \"Aug 28, 2013\"
10.SH NAME
11 lofiadm \- administer files available as block devices through lofi
12.SH SYNOPSIS
13.LP
14.nf
15 \fBlofiadm\fR [\fB-r\fR] \fB-a\fR \fIfile\fR [\fIdevice\fR]
16.fi

18.LP
19.nf
20 \fBlofiadm\fR [\fB-r\fR] [\fB-o\fR] \fB-c\fR \fIcrypto_algorithm\fR \fB-a\fR \fIfI
2 \fBlofiadm\fR [\fB-r\fR] \fB-c\fR \fIcrypto_algorithm\fR \fB-a\fR \fIfile\fR [\fI
21.fi

23.LP
24.nf
25 \fBlofiadm\fR [\fB-r\fR] \fB-c\fR \fIcrypto_algorithm\fR \fB-k\fR \fIraw_key_fil
26.fi

28.LP
29.nf
30 \fBlofiadm\fR [\fB-r\fR] \fB-c\fR \fIcrypto_algorithm\fR \fB-T\fR \fItoken_key\f
31.fi

33.LP
34.nf
35 \fBlofiadm\fR [\fB-r\fR] \fB-c\fR \fIcrypto_algorithm\fR \fB-T\fR \fItoken_key\f
36 \fB-k\fR \fIwrapped_key_file\fR \fB-a\fR \fIfile\fR [\fIdevice\fR]
37.fi

39.LP
40.nf
41 \fBlofiadm\fR [\fB-r\fR] \fB-c\fR \fIcrypto_algorithm\fR \fB-e\fR \fB-a\fR \fIfi
42.fi

44.LP
45.nf
46 \fBlofiadm\fR \fB-C\fR \fIalgorithm\fR [\fB-s\fR \fIsegment_size\fR] \fIfile\fR
47.fi

49.LP
50.nf
51 \fBlofiadm\fR \fB-d\fR \fIfile\fR | \fIdevice\fR
52.fi

54.LP
55.nf
56 \fBlofiadm\fR \fB-U\fR \fIfile\fR
57.fi

59.LP
60.nf
```

new/usr/src/man/man1m/lofiadm.1m

2

```
61 \fBlofiadm\fR [ \fIfile\fR | \fIdevice\fR]
62.fi

64.SH DESCRIPTION
65.sp
66 \fBlofiadm\fR administers \fBblofi\fR, the loopback file driver. \fBblofi\fR
67 allows a file to be associated with a block device. That file can then be
68 accessed through the block device. This is useful when the file contains an
69 image of some filesystem (such as a floppy or \fBCD-ROM\fR image), because the
70 block device can then be used with the normal system utilities for mounting,
71 checking or repairing filesystems. See \fBfsck\fR(1M) and \fBmount\fR(1M).
72.sp
73.LP
74 Use \fBlofiadm\fR to add a file as a loopback device, remove such an
75 association, or print information about the current associations.
76.sp
77.LP
78 Encryption and compression options are mutually exclusive on the command line.
79 Further, an encrypted file cannot be compressed later, nor can a compressed
80 file be encrypted later.

82 In the global zone, \fBlofiadm\fR can be used on both the global
83 zone devices and all devices owned by other non-global zones on the system.
84.sp
85.SH OPTIONS
86.sp
87 The following options are supported:
88.sp
89.ne 2
90.na
91 \fB-a\fR \fIfile\fR [\fIdevice\fR]\fR
92.ad
93.sp .6
94.RS 4n
95 Add \fIfile\fR as a block device.
96.sp
97 If \fIdevice\fR is not specified, an available device is picked.
98.sp
99 If \fIdevice\fR is specified, \fBlofiadm\fR attempts to assign it to
100 \fIfile\fR. \fIdevice\fR must be available or \fBlofiadm\fR will fail. The
101 ability to specify a device is provided for use in scripts that wish to
102 reestablish a particular set of associations.
103.RE

105.sp
106.ne 2
107.na
108 \fB-C\fR {\fIgzip\fR | \fIgzip-N\fR | \fIlzma\fR}\fR
109.ad
110.sp .6
111.RS 4n
112 Compress the file with the specified compression algorithm.
113.sp
114 The \fBgzip\fR compression algorithm uses the same compression as the
115 open-source \fBgzip\fR command. You can specify the \fBgzip\fR level by using
116 the value \fBgzip-\fR\fIN\fR where \fIN\fR is 6 (fast) or 9 (best compression
117 ratio). Currently, \fBgzip\fR, without a number, is equivalent to \fBgzip-6\fR
118 (which is also the default for the \fBgzip\fR command).
119.sp
120 \fIlzma\fR stands for the LZMA (Lempel-Ziv-Markov) compression algorithm.
121.sp
122 Note that you cannot write to a compressed file, nor can you mount a compressed
123 file read/write.
124.RE
```

```

126 .sp
127 .ne 2
128 .na
129 \fB\fB-d\fR \fIfile\fR | \fIdevice\fR\fR
130 .ad
131 .sp .6
132 .RS 4n
133 Remove an association by \fIfile\fR or \fIdevice\fR name, if the associated
134 block device is not busy, and deallocates the block device.
135 .RE

137 .sp
138 .ne 2
139 .na
140 \fB\fB-o\fR
141 .ad
142 .sp .6
143 .RS 4n
144 If the \fB-o\fR option is specified lofiadm will prompt for a passphrase once.
145 .RE

147 .sp
148 .ne 2
149 .na
150 #endif /* ! codereview */
151 \fB\fB-r\fR
152 .ad
153 .sp .6
154 .RS 4n
155 If the \fB-r\fR option is specified before the \fB-a\fR option, the
156 \fIdevice\fR will be opened read-only.
157 .RE

159 .sp
160 .ne 2
161 .na
162 \fB\fB-s\fR \fIsegment_size\fR\fR
163 .ad
164 .sp .6
165 .RS 4n
166 The segment size to use to divide the file being compressed. \fIsegment_size\fR
167 can be an integer multiple of 512.
168 .RE

170 .sp
171 .ne 2
172 .na
173 \fB\fB-U\fR \fIfile\fR\fR
174 .ad
175 .sp .6
176 .RS 4n
177 Uncompress a compressed file.
178 .RE

180 .sp
181 .LP
182 The following options are used when the file is encrypted:
183 .sp
184 .ne 2
185 .na
186 \fB\fB-c\fR \fIcrypto_algorithm\fR\fR
187 .ad
188 .sp .6
189 .RS 4n
190 Select the encryption algorithm. The algorithm must be specified when

```

```

191 encryption is enabled because the algorithm is not stored in the disk image.
192 .sp
193 If none of \fB-e\fR, \fB-k\fR, or \fB-T\fR is specified, \fBlofiadm\fR prompts
194 for a passphrase, with a minimum length of eight characters, to be entered .
195 The passphrase is used to derive a symmetric encryption key using PKCS#5 PBKDF2.
196 .RE

198 .sp
199 .ne 2
200 .na
201 \fB\fB-k\fR \fIraw_key_file\fR | \fIwrapped_key_file\fR\fR
202 .ad
203 .sp .6
204 .RS 4n
205 Path to raw or wrapped symmetric encryption key. If a PKCS#11 object is also
206 given with the \fB-T\fR option, then the key is wrapped by that object. If
207 \fB-T\fR is not specified, the key is used raw.
208 .RE

210 .sp
211 .ne 2
212 .na
213 \fB\fB-T\fR \fItoken_key\fR\fR
214 .ad
215 .sp .6
216 .RS 4n
217 The key in a PKCS#11 token to use for the encryption or for unwrapping the key
218 file.
219 .sp
220 If \fB-k\fR is also specified, \fB-T\fR identifies the unwrapping key, which
221 must be an RSA private key.
222 .RE

224 .sp
225 .ne 2
226 .na
227 \fB\fB-e\fR\fR
228 .ad
229 .sp .6
230 .RS 4n
231 Generate an ephemeral symmetric encryption key.
232 .RE

234 .SH OPERANDS
235 .sp
236 .LP
237 The following operands are supported:
238 .sp
239 .na
240 \fB\fIcrypto_algorithm\fR\fR
241 .ad
242 .sp .6
243 .RS 4n
244 One of: \fBbaes-128-cbc\fR, \fBbaes-192-cbc\fR, \fBbaes-256-cbc\fR,
245 \fBdes3-cbc\fR, \fBblowfish-cbc\fR.
246 .RE

248 .sp
249 .ne 2
250 .na
251 \fB\fIdevice\fR\fR
252 .ad
253 .sp .6
254 .RS 4n
255 Display the file name associated with the block device \fIdevice\fR.

```

```

256 .sp
257 Without arguments, print a list of the current associations. Filenames must be
258 valid absolute pathnames.
259 .sp
260 When a file is added, it is opened for reading or writing by root. Any
261 restrictions apply (such as restricted root access over \fBNFS\fR). The file is
262 held open until the association is removed. It is not actually accessed until
263 the block device is used, so it will never be written to if the block device is
264 only opened read-only.

```

```

266 Note that the filename may appear as "?" if it is not possible to resolve the
267 path in the current context (for example, if it's an NFS path in a non-global
268 zone).
269 .RE

```

```

271 .sp
272 .ne 2
273 .na
274 \fB\fIfile\fR\fR
275 .ad
276 .sp .6
277 .RS 4n
278 Display the block device associated with \fIfile\fR.
279 .RE

```

```

281 .sp
282 .ne 2
283 .na
284 \fB\fIRaw_key_file\fR\fR
285 .ad
286 .sp .6
287 .RS 4n
288 Path to a file of the appropriate length, in bits, to use as a raw symmetric
289 encryption key.
290 .RE

```

```

292 .sp
293 .ne 2
294 .na
295 \fB\fIToken_key\fR\fR
296 .ad
297 .sp .6
298 .RS 4n
299 PKCS#11 token object in the format:
300 .sp
301 .in +2
302 .nf
303 \fIToken_name\fR:\fIManufacturer_id\fR:\fISerial_number\fR:\fIkey_label\fR
304 .fi
305 .in -2
306 .sp

```

```

308 All but the key label are optional and can be empty. For example, to specify a
309 token object with only its key label \fBMylofiKey\fR, use:

```

```

310 .sp
311 .in +2
312 .nf
313 -T ::MylofiKey
314 .fi
315 .in -2
316 .sp

```

```

318 .RE

```

```

320 .sp
321 .ne 2

```

```

322 .na
323 \fB\fIwrapped_key_file\fR\fR
324 .ad
325 .sp .6
326 .RS 4n
327 Path to file containing a symmetric encryption key wrapped by the RSA private
328 key specified by \fB-T\fR.
329 .RE

```

```

331 .SH EXAMPLES
332 .LP
333 \fBExample 1 \fRMounting an Existing CD-ROM Image
334 .sp
335 .LP
336 You should ensure that Solaris understands the image before creating the
337 \fBCD\fR. \fBlofi\fR allows you to mount the image and see if it works.

```

```

339 .sp
340 .LP
341 This example mounts an existing \fBCD-ROM\fR image (\fBsparc.iso\fR), of the
342 \fBRed Hat 6.0 CD\fR which was downloaded from the Internet. It was created
343 with the \fBmkisofs\fR utility from the Internet.

```

```

345 .sp
346 .LP
347 Use \fBlofiadm\fR to attach a block device to it:

```

```

349 .sp
350 .in +2
351 .nf
352 # \fBlofiadm -a /home/mike_s/RH6.0/sparc.iso\fR
353 /dev/lofi/1
354 .fi
355 .in -2
356 .sp

```

```

358 .sp
359 .LP
360 \fBlofiadm\fR picks the device and prints the device name to the standard
361 output. You can run \fBlofiadm\fR again by issuing the following command:

```

```

363 .sp
364 .in +2
365 .nf
366 # \fBlofiadm\fR
367 Block Device      File                      Options
368 /dev/lofi/1      /home/mike_s/RH6.0/sparc.iso  -
369 .fi
370 .in -2
371 .sp

```

```

373 .sp
374 .LP
375 Or, you can give it one name and ask for the other, by issuing the following
376 command:

```

```

378 .sp
379 .in +2
380 .nf
381 # \fBlofiadm /dev/lofi/1\fR
382 /home/mike_s/RH6.0/sparc.iso
383 .fi
384 .in -2
385 .sp

```

```

387 .sp

```

```

388 .LP
389 Use the \fBmount\fR command to mount the image:

391 .sp
392 .in +2
393 .nf
394 # \fBmount -F hsfs -o ro /dev/lofi/1 /mnt\fR
395 .fi
396 .in -2
397 .sp

399 .sp
400 .LP
401 Check to ensure that Solaris understands the image:

403 .sp
404 .in +2
405 .nf
406 # \fBdf -k /mnt\fR
407 Filesystem          kbytes    used    avail capacity  Mounted on
408 /dev/lofi/1          512418    512418         0    100%      /mnt
409 # \fBls /mnt\fR
410 \&./                RedHat/      doc/         ls-lR         rr_moved/
411 \&../                TRANS.TBL    dosutils/    ls-lR.gz      sbin@
412 \&.buildlog         bin@         etc@         misc/         tmp/
413 COPYING             boot/        images/      mnt/          usr@
414 README              boot.cat*    kernels/     modules/
415 RPM-PGP-KEY         dev@         lib@         proc/
416 .fi
417 .in -2
418 .sp

420 .sp
421 .LP
422 Solaris can mount the CD-ROM image, and understand the filenames. The image was
423 created properly, and you can now create the \fBCD-ROM\fR with confidence.

425 .sp
426 .LP
427 As a final step, unmount and detach the images:

429 .sp
430 .in +2
431 .nf
432 # \fBumount /mnt\fR
433 # \fBblofiadm -d /dev/lofi/1\fR
434 # \fBblofiadm\fR
435 Block Device          File          Options
436 .fi
437 .in -2
438 .sp

440 .LP
441 \fBExample 2 \fRMounting a Floppy Image
442 .sp
443 .LP
444 This is similar to the first example.

446 .sp
447 .LP
448 Using \fBblofi\fR to help you mount files that contain floppy images is helpful
449 if a floppy disk contains a file that you need, but the machine which you are
450 on does not have a floppy drive. It is also helpful if you do not want to take
451 the time to use the \fBdd\fR command to copy the image to a floppy.

453 .sp

```

```

454 .LP
455 This is an example of getting to \fBMBD\fR floppy for Solaris on an x86
456 platform:

458 .sp
459 .in +2
460 .nf
461 # \fBblofiadm -a /export/s28/MBD_s28x_wos/latest/boot.3\fR
462 /dev/lofi/1
463 # \fBmount -F pcfs /dev/lofi/1 /mnt\fR
464 # \fBls /mnt\fR
465 \&./                COMMENT.BAT*  RC.D/        SOLARIS.MAP*
466 \&../                IDENT*        REPLACE.BAT*  X/
467 APPEND.BAT*        MAKEDIR.BAT*  SOLARIS/
468 # \fBumount /mnt\fR
469 # \fBblofiadm -d /export/s28/MBD_s28x_wos/latest/boot.3\fR
470 .fi
471 .in -2
472 .sp

474 .LP
475 \fBExample 3 \fRMaking a \fBUFS\fR Filesystem on a File
476 .sp
477 .LP
478 Making a \fBUFS\fR filesystem on a file can be useful, particularly if a test
479 suite requires a scratch filesystem. It can be painful (or annoying) to have to
480 repartition a disk just for the test suite, but you do not have to. You can
481 \fBnewfs\fR a file with \fBblofi\fR

483 .sp
484 .LP
485 Create the file:

487 .sp
488 .in +2
489 .nf
490 # \fBmkfile 35m /export/home/test\fR
491 .fi
492 .in -2
493 .sp

495 .sp
496 .LP
497 Attach it to a block device. You also get the character device that \fBnewfs\fR
498 requires, so \fBnewfs\fR that:

500 .sp
501 .in +2
502 .nf
503 # \fBblofiadm -a /export/home/test\fR
504 /dev/lofi/1
505 # \fBnewfs /dev/rlofi/1\fR
506 newfs: construct a new file system /dev/rlofi/1: (y/n)? \fBy\fR
507 /dev/rlofi/1: 71638 sectors in 119 cylinders of 1 tracks, 602 sectors
508 35.0MB in 8 cyl groups (16 c/g, 4.70MB/g, 2240 i/g)
509 super-block backups (for fsck -F ufs -o b=#) at:
510 32, 9664, 19296, 28928, 38560, 48192, 57824, 67456,
511 .fi
512 .in -2
513 .sp

515 .sp
516 .LP
517 Note that \fBbufs\fR might not be able to use the entire file. Mount and use the
518 filesystem:

```



```

520 .sp
521 .in +2
522 .nf
523 # \fBmount /dev/lofi/1 /mnt\fR
524 # \fBdf -k /mnt\fR
525 Filesystem          kbytes    used    avail capacity  Mounted on
526 /dev/lofi/1          33455      9    30101      1%    /mnt
527 # \fBls /mnt\fR
528 \&./                ../      lost+found/
529 # \fBumount /mnt\fR
530 # \fBlofiadm -d /dev/lofi/1\fR
531 .fi
532 .in -2
533 .sp

535 .LP
536 \fBExample 4 \fRCreating a PC (FAT) File System on a Unix File
537 .sp
538 .LP
539 The following series of commands creates a \fBFAT\fR file system on a Unix
540 file. The file is associated with a block device created by \fBlofiadm\fR.

542 .sp
543 .in +2
544 .nf
545 # \fBmkfile 10M /export/test/testfs\fR
546 # \fBlofiadm -a /export/test/testfs\fR
547 /dev/lofi/1
548 \fBNote use of\fR rlofi\fB, not\fR lofi\fB, in following command.\fR
549 # \fBmkfs -F pcfs -o nofdisk,size=20480 /dev/rlofi/1\fR
550 \fBConstruct a new FAT file system on /dev/rlofi/1: (y/n)?\fR y
551 # \fBmount -F pcfs /dev/lofi/1 /mnt\fR
552 # \fBcd /mnt\fR
553 # \fBdf -k .\fR
554 Filesystem          kbytes    used    avail capacity  Mounted on
555 /dev/lofi/1          10142      0    10142      0%    /mnt
556 .fi
557 .in -2
558 .sp

560 .LP
561 \fBExample 5 \fRCompressing an Existing CD-ROM Image
562 .sp
563 .LP
564 The following example illustrates compressing an existing CD-ROM image
565 (\fBsolaris.iso\fR), verifying that the image is compressed, and then
566 uncompressing it.

568 .sp
569 .in +2
570 .nf
571 # \fBlofiadm -C gzip /export/home/solaris.iso\fR
572 .fi
573 .in -2
574 .sp

576 .sp
577 .LP
578 Use \fBlofiadm\fR to attach a block device to it:

580 .sp
581 .in +2
582 .nf
583 # \fBlofiadm -a /export/home/solaris.iso\fR
584 /dev/lofi/1
585 .fi

```

```

586 .in -2
587 .sp

589 .sp
590 .LP
591 Check if the mapped image is compressed:

593 .sp
594 .in +2
595 .nf
596 # \fBlofiadm\fR
597 Block Device      File      Options
598 /dev/lofi/1      /export/home/solaris.iso  Compressed(gzip)
599 /dev/lofi/2      /export/home/regular.iso  -
600 .fi
601 .in -2
602 .sp

604 .sp
605 .LP
606 Unmap the compressed image and uncompress it:

608 .sp
609 .in +2
610 .nf
611 # \fBlofiadm -d /dev/lofi/1\fR
612 # \fBlofiadm -U /export/home/solaris.iso\fR
613 .fi
614 .in -2
615 .sp

617 .LP
618 \fBExample 6 \fRCreating an Encrypted UFS File System on a File
619 .sp
620 .LP
621 This example is similar to the example of making a UFS filesystem on a file,
622 above.

624 .sp
625 .LP
626 Create the file:

628 .sp
629 .in +2
630 .nf
631 # \fBmkfile 35m /export/home/test\fR
632 .fi
633 .in -2
634 .sp

636 .sp
637 .LP
638 Attach the file to a block device and specify that the file image is encrypted.
639 As a result of this command, you obtain the character device, which is
640 subsequently used by \fBnewfs\fR:

642 .sp
643 .in +2
644 .nf
645 # \fBlofiadm -c aes-256-cbc -a /export/home/secrets\fR
646 Enter passphrase: \fBMy-M0th3r;l0v3s_m3+4lw4ys!\fR      (\fBnot echoed\fR)
647 Re-enter passphrase: \fBMy-M0th3r;l0v3s_m3+4lw4ys!\fR      (\fBnot echoed\fR)
648 /dev/lofi/1

650 # \fBnewfs /dev/rlofi/1\fR
651 newfs: construct a new file system /dev/rlofi/1: (y/n)? \fBy\fR

```

```

652 /dev/rlofi/1: 71638 sectors in 119 cylinders of 1 tracks, 602 sectors
653 35.0MB in 8 cyl groups (16 c/g, 4.70MB/g, 2240 i/g)
654 super-block backups (for fsck -F ufs -o b=#) at:
655 32, 9664, 19296, 28928, 38560, 48192, 57824, 67456,
656 .fi
657 .in -2
658 .sp

660 .sp
661 .LP
662 The mapped file system shows that encryption is enabled:

664 .sp
665 .in +2
666 .nf
667 # \fBlofiadm\fR
668 Block Device File Options
669 /dev/lofi/1 /export/home/secrets Encrypted
670 .fi
671 .in -2
672 .sp

674 .sp
675 .LP
676 Mount and use the filesystem:

678 .sp
679 .in +2
680 .nf
681 # \fBmount /dev/lofi/1 /mnt\fR
682 # \fBcp moms_secret_*_recipe /mnt\fR
683 # \fBls /mnt\fR
684 \&./ moms_secret_cookie_recipe moms_secret_soup_recipe
685 \&./ moms_secret_fudge_recipe moms_secret_stuffing_recipe
686 lost+found/ moms_secret_meatloaf_recipe moms_secret_waffle_recipe
687 # \fBumount /mnt\fR
688 # \fBlofiadm -d /dev/lofi/1\fR
689 .fi
690 .in -2
691 .sp

693 .sp
694 .LP
695 Subsequent attempts to map the filesystem with the wrong key or the wrong
696 encryption algorithm will fail:

698 .sp
699 .in +2
700 .nf
701 # \fBlofiadm -c blowfish-cbc -a /export/home/secrets\fR
702 Enter passphrase: \fBmommy\fR (\fInot echoed\fR)
703 Re-enter passphrase: \fBmommy\fR (\fInot echoed\fR)
704 lofiadm: could not map file /root/lofi: Invalid argument
705 # \fBlofiadm\fR
706 Block Device File Options
707 #
708 .fi
709 .in -2
710 .sp

712 .sp
713 .LP
714 Attempts to map the filesystem without encryption will succeed, however
715 attempts to mount and use the filesystem will fail:

717 .sp

```

```

718 .in +2
719 .nf
720 # \fBlofiadm -a /export/home/secrets\fR
721 /dev/lofi/1
722 # \fBlofiadm\fR
723 Block Device File Options
724 /dev/lofi/1 /export/home/secrets -
725 # \fBmount /dev/lofi/1 /mnt\fR
726 mount: /dev/lofi/1 is not this fstype
727 #
728 .fi
729 .in -2
730 .sp

732 .SH ENVIRONMENT VARIABLES
733 .sp
734 .LP
735 See \fBenvron\fR(5) for descriptions of the following environment variables
736 that affect the execution of \fBlofiadm\fR: \fBLC_CTYPE\fR, \fBLC_MESSAGES\fR
737 and \fBNLSPATH\fR.
738 .SH EXIT STATUS
739 .sp
740 .LP
741 The following exit values are returned:
742 .sp
743 .ne 2
744 .na
745 \fB0\fR
746 .ad
747 .sp .6
748 .RS 4n
749 Successful completion.
750 .RE

752 .sp
753 .ne 2
754 .na
755 \fB>0\fR
756 .ad
757 .sp .6
758 .RS 4n
759 An error occurred.
760 .RE

762 .SH SEE ALSO
763 .sp
764 .LP
765 \fBfsck\fR(1M), \fBmount\fR(1M), \fBmount_ufs\fR(1M), \fBnewfs\fR(1M),
766 \fBattributes\fR(5), \fBlofi\fR(7D), \fBlofs\fR(7FS)
767 .SH NOTES
768 .sp
769 .LP
770 Just as you would not directly access a disk device that has mounted file
771 systems, you should not access a file associated with a block device except
772 through the \fBlofi\fR file driver. It might also be appropriate to ensure that
773 the file has appropriate permissions to prevent such access.
774 .sp
775 .LP
776 The abilities of \fBlofiadm\fR, and who can use them, are controlled by the
777 permissions of \fB/dev/lofi1\fR. Read-access allows query operations, such as
778 listing all the associations. Write-access is required to do any state-changing
779 operations, like adding an association. As shipped, \fB/dev/lofi1\fR is owned
780 by \fBroot\fR, in group \fBsys\fR, and mode \fB0644\fR, so all users can do
781 query operations but only root can change anything. The administrator can give
782 users write-access, allowing them to add or delete associations, but that is
783 very likely a security hole and should probably only be given to a trusted

```

780 group.
781 .sp
782 .LP
783 When mounting a filesystem image, take care to use appropriate mount options.
784 In particular, the \fBnosuid\fR mount option might be appropriate for \fBUFS\fR
785 images whose origin is unknown. Also, some options might not be useful or
786 appropriate, like \fBlogging\fR or \fBforcedirectio\fR for \fBUFS\fR. For
787 compatibility purposes, a raw device is also exported along with the block
788 device. For example, \fBnewfs\fR(1M) requires one.
789 .sp
790 .LP
791 The output of \fBlofiadm\fR (without arguments) might change in future
792 releases.