```
**********************************************************
    7952 Mon Jan 21 16:25:58 2019
new/usr/src/cmd/cmd-crypto/kmfcfg/kmfcfg.c
10126 smatch fix for kmfcfg
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 *
 * Copyright (c) 2006, 2010, Oracle and/or its affiliates. All rights reserved.
 */

/*
 * Copyright (c) 2018, Joyent, Inc.
 */

#include <stdio.h>
#include <strings.h>
#include <ctype.h>
#include <libgen.h>
#include <libintl.h>
#include <locale.h>

#include <kmfapiP.h>

#include "util.h"

/*
 * The verbcmd construct allows genericizing information about a verb so
 * that it is easier to manipulate.  Makes parsing code easier to read,
 * fix, and extend with new verbs.
 */
typedef struct verbcmd_s {
        char            *verb;
        int             (*action)(int, char *[]);
        char            *synopsis;
} verbcmd;
______unchanged_portion_omitted_

int
main(int argc, char *argv[])
{
        int ret;
        int found;
        int i;

        (void) setlocale(LC_ALL, "");
#if !defined(TEXT_DOMAIN)               /* Should be defined by cc -D. */
#define TEXT_DOMAIN     "SYS_TEST"      /* Use this only if it isn't. */
#endif
```

```
        (void) textdomain(TEXT_DOMAIN);

        prog = basename(argv[0]);
        argv++;
        argc--;
        argv++; argc--;

        if (argc == 0) {
                usage();
                exit(1);
        }

        if (argc == 1 && argv[0][0] == '-') {
                switch (argv[0][1]) {
                        case '?':
                                return (kc_help());
                        default:
                                usage();
                                exit(1);
                }
        }

        found = -1;
        for (i = 0; i < num_cmds; i++) {
                if (strcmp(cmds[i].verb, argv[0]) == 0) {
                        found = i;
                        break;
                }
        }

        if (found < 0) {
                (void) fprintf(stderr, gettext("Invalid command: %s\n"),
                    argv[0]);
                exit(1);
        }

        /*
         * Note the action functions can return values from
         * the key management framework, and those values can conflict
         * with the utility error codes.
         */
        ret = (*cmds[found].action)(argc, argv);

        switch (ret) {
                case KC_OK:
                        break;
                case KC_ERR_USAGE:
                        break;
                case KC_ERR_LOADDB:
                        (void) fprintf(stderr,
                            gettext("Error loading database\n"));
                        break;
                case KC_ERR_FIND_POLICY:
                        break;
                case KC_ERR_DELETE_POLICY:
                        (void) fprintf(stderr, gettext("Error deleting policy "
                            "from database.\n"));
                        break;
                case KC_ERR_ADD_POLICY:
                        break;
                case KC_ERR_VERIFY_POLICY:
                        break;
                case KC_ERR_INCOMPLETE_POLICY:
                        break;
                case KC_ERR_MEMORY:
                        (void) fprintf(stderr, gettext("Out of memory.\n"));
                        break;
                case KC_ERR_ACCESS:
                        break;
                case KC_ERR_INSTALL:
                        break;
                case KC_ERR_UNINSTALL:
                        break;
                default:
                        (void) fprintf(stderr, gettext("%s operation failed. "
                            "error 0x%02x\n"), cmds[found].verb, ret);
                        break;
        }

        return (ret);
}
```

______***unchanged_portion_omitted_***