```
**********************************************************
    7959 Thu Jan 24 09:47:47 2019
new/usr/src/cmd/fm/fmdump/common/fault.c
10130 smatch fixes for usr/src/cmd/fm
**********************************************************
```

```
 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 (c) 2004, 2010, Oracle and/or its affiliates. All rights reserved.
23  * Copyright (c) 2018, Joyent, Inc.
23  * Copyright (c) 2017, Joyent, Inc. All rights reserved.
24  */

26 #include <fmdump.h>
27 #include <stdio.h>
28 #include <strings.h>

30 /*ARGSUSED*/
31 static int
32 flt_short(fmd_log_t *lp, const fmd_log_record_t *rp, FILE *fp)
33 {
34         char buf[32], str[32];
35         char *class = NULL, *uuid = "-", *code = "-";

37         static const struct {
38                 const char *class;
39                 const char *tag;
40         } tags[] = {
41                 { FM_LIST_SUSPECT_CLASS,        "Diagnosed" },
42                 { FM_LIST_REPAIRED_CLASS,       "Repaired" },
43                 { FM_LIST_RESOLVED_CLASS,       "Resolved" },
44                 { FM_LIST_UPDATED_CLASS,        "Updated" },
45                 { FM_LIST_ISOLATED_CLASS,       "Isolated" },
46         };

48         (void) nvlist_lookup_string(rp->rec_nvl, FM_SUSPECT_UUID, &uuid);
49         (void) nvlist_lookup_string(rp->rec_nvl, FM_SUSPECT_DIAG_CODE, &code);

51         (void) nvlist_lookup_string(rp->rec_nvl, FM_CLASS, &class);
52         if (class != NULL) {
53                 int i;

55                 for (i = 0; i < sizeof (tags) / sizeof (tags[0]); i++) {
56                         if (strcmp(class, tags[i].class) == 0) {
57                                 (void) snprintf(str, sizeof (str), "%s %s",
58                                     code, tags[i].tag);
59                                 code = str;
60                                 break;
```



```
61                         }
62                 }
63         }

65         fmdump_printf(fp, "%-20s %-32s %s\n",
66             fmdump_date(buf, sizeof (buf), rp), uuid, code);

68         return (0);
69 }
______unchanged_portion_omitted_

150 static int
151 flt_verb23_cmn(fmd_log_t *lp, const fmd_log_record_t *rp, FILE *fp,
152     nvlist_prtctl_t pctl)
153 {
154         const struct fmdump_fmt *efp = &fmdump_err_ops.do_formats[FMDUMP_VERB1];
155         const struct fmdump_fmt *ffp = &fmdump_flt_ops.do_formats[FMDUMP_VERB2];
156         uint_t i;
157         char buf[32], str[32];
158         char *class = NULL, *uuid = "-", *code = "-";

160         (void) nvlist_lookup_string(rp->rec_nvl, FM_SUSPECT_UUID, &uuid);
161         (void) nvlist_lookup_string(rp->rec_nvl, FM_SUSPECT_DIAG_CODE, &code);

163         (void) nvlist_lookup_string(rp->rec_nvl, FM_CLASS, &class);
164         if (class != NULL && strcmp(class, FM_LIST_REPAIRED_CLASS) == 0) {
165                 (void) snprintf(str, sizeof (str), "%s %s", code, "Repaired");
166                 code = str;
167         }
168         if (class != NULL && strcmp(class, FM_LIST_RESOLVED_CLASS) == 0) {
169                 (void) snprintf(str, sizeof (str), "%s %s", code, "Resolved");
170                 code = str;
171         }
172         if (class != NULL && strcmp(class, FM_LIST_UPDATED_CLASS) == 0) {
173                 (void) snprintf(str, sizeof (str), "%s %s", code, "Updated");
174                 code = str;
175         }

177         fmdump_printf(fp, "%s\n", ffp->do_hdr);
178         fmdump_printf(fp, "%-20s.%9.9llu %-32s %s\n",
179             fmdump_year(buf, sizeof (buf), rp), rp->rec_nsec, uuid, code);

181         if (rp->rec_nrefs != 0)
182                 fmdump_printf(fp, "\n  %s\n", efp->do_hdr);

184         for (i = 0; i < rp->rec_nrefs; i++) {
185                 fmdump_printf(fp, "  ");
186                 (void) efp->do_func(lp, &rp->rec_xrefs[i], fp);
186                 efp->do_func(lp, &rp->rec_xrefs[i], fp);
187         }

189         fmdump_printf(fp, "\n");
190         if (pctl)
191                 nvlist_prt(rp->rec_nvl, pctl);
192         else
193                 nvlist_print(fp, rp->rec_nvl);
194         fmdump_printf(fp, "\n");

196         return (0);
197 }
______unchanged_portion_omitted_
```

```
**********************************************************
   26401 Thu Jan 24 09:47:47 2019
new/usr/src/cmd/fm/notify/smtp-notify/common/smtp-notify.c
10130 smatch fixes for usr/src/cmd/fm
**********************************************************
```

```
  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  */

 22 /*
 23  * Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
 24  */

 26 /*
 27  * Copyright (c) 2018, Joyent, Inc.
 28  */

 30 #include <stdio.h>
 31 #include <stdlib.h>
 32 #include <string.h>
 33 #include <alloca.h>
 34 #include <errno.h>
 35 #include <fcntl.h>
 36 #include <libscf.h>
 37 #include <priv_utils.h>
 38 #include <netdb.h>
 39 #include <signal.h>
 40 #include <strings.h>
 41 #include <time.h>
 42 #include <unistd.h>
 43 #include <zone.h>
 44 #include <sys/types.h>
 45 #include <sys/stat.h>
 46 #include <fm/fmd_msg.h>
 47 #include <fm/libfmevent.h>
 48 #include "libfmnotify.h"

 50 #define SENDMAIL        "/usr/sbin/sendmail"
 51 #define SVCNAME         "system/fm/smtp-notify"

 53 #define XHDR_HOSTNAME           "X-FMEV-HOSTNAME"
 54 #define XHDR_CLASS              "X-FMEV-CLASS"
 55 #define XHDR_UUID               "X-FMEV-UUID"
 56 #define XHDR_MSGID              "X-FMEV-CODE"
 57 #define XHDR_SEVERITY           "X-FMEV-SEVERITY"
 58 #define XHDR_FMRI               "X-FMEV-FMRI"
 59 #define XHDR_FROM_STATE         "X-FMEV-FROM-STATE"
 60 #define XHDR_TO_STATE           "X-FMEV-TO-STATE"
```



```
 62 /*
 63  * Debug messages can be enabled by setting the debug property to true
 64  *
 65  * # svccfg -s svc:/system/fm/smtp-notify setprop config/debug=true
 66  *
 67  * Debug messages will be spooled to the service log at:
 68  * <root>/var/svc/log/system-fm-smtp-notify:default.log
 69  */
 70 #define PP_SCRIPT "usr/lib/fm/notify/process_msg_template.sh"

 72 typedef struct email_pref
 73 {
 74         int ep_num_recips;
 75         char **ep_recips;
 76         char *ep_reply_to;
 77         char *ep_template_path;
 78         char *ep_template;
 79 } email_pref_t;
_____unchanged_portion_omitted_

779 int
780 main(int argc, char *argv[])
781 {
782         struct rlimit rlim;
783         struct sigaction act;
784         sigset_t set;
785         char c;
786         boolean_t run_fg = B_FALSE;

788         if ((nhdl = malloc(sizeof (nd_hdl_t))) == NULL) {
789                 (void) fprintf(stderr, "Failed to allocate space for notifyd "
790                     "handle (%s)", strerror(errno));
791                 return (1);
792         }
793         (void) memset(nhdl, 0, sizeof (nd_hdl_t));

795         nhdl->nh_keep_running = B_TRUE;
796         nhdl->nh_log_fd = stderr;
797         nhdl->nh_pname = argv[0];

799         get_svc_config();

801         /*
802          * In the case where we get started outside of SMF, args passed on the
803          * command line override SMF property setting
804          */
805         while (optind < argc) {
806                 while ((c = getopt(argc, argv, optstr)) != -1) {
807                         switch (c) {
808                         case 'd':
809                                 nhdl->nh_debug = B_TRUE;
810                                 break;
811                         case 'f':
812                                 run_fg = B_TRUE;
813                                 break;
814                         case 'R':
815                                 nhdl->nh_rootdir = strdup(optarg);
816                                 break;
817                         default:
818                                 free(nhdl);
819                                 return (usage(argv[0]));
814                                 return (usage(nhdl->nh_pname));
820                         }
821                 }
822         }
```

```
824         /*
825          * Set up a signal handler for SIGTERM (and SIGINT if we'll
826          * be running in the foreground) to ensure sure we get a chance to exit
827          * in an orderly fashion.  We also catch SIGHUP, which will be sent to
828          * us by SMF if the service is refreshed.
829          */
830         (void) sigfillset(&set);
831         (void) sigfillset(&act.sa_mask);
832         act.sa_handler = nd_sighandler;
833         act.sa_flags = 0;

835         (void) sigaction(SIGTERM, &act, NULL);
836         (void) sigdelset(&set, SIGTERM);
837         (void) sigaction(SIGHUP, &act, NULL);
838         (void) sigdelset(&set, SIGHUP);

840         if (run_fg) {
841                 (void) sigaction(SIGINT, &act, NULL);
842                 (void) sigdelset(&set, SIGINT);
843         } else
844                 nd_daemonize(nhdl);

846         rlim.rlim_cur = RLIM_INFINITY;
847         rlim.rlim_max = RLIM_INFINITY;
848         (void) setrlimit(RLIMIT_CORE, &rlim);

850         /*
851          * We need to be root to initialize our libfmevent handle (because that
852          * involves reading/writing to /dev/sysevent), so we do this before
853          * calling __init_daemon_priv.
854          */
855         nhdl->nh_evhdl = fmev_shdl_init(LIBFMEVENT_VERSION_2, NULL, NULL, NULL);
856         if (nhdl->nh_evhdl == NULL) {
857                 (void) sleep(5);
858                 nd_abort(nhdl, "failed to initialize libfmevent: %s",
859                     fmev_strerror(fmev_errno));
860         }

862         /*
863          * If we're in the global zone, reset all of our privilege sets to
864          * the minimum set of required privileges.  Since we've already
865          * initialized our libmevent handle, we no no longer need to run as
866          * root, so we change our uid/gid to noaccess (60002).
867          *
868          * __init_daemon_priv will also set the process core path for us
869          *
870          */
871         if (getzoneid() == GLOBAL_ZONEID)
872                 if (__init_daemon_priv(
873                     PU_RESETGROUPS | PU_LIMITPRIVS | PU_INHERITPRIVS,
874                     60002, 60002, PRIV_PROC_SETID, NULL) != 0)
875                         nd_abort(nhdl, "additional privileges required to run");

877         nhdl->nh_msghdl = fmd_msg_init(nhdl->nh_rootdir, FMD_MSG_VERSION);
878         if (nhdl->nh_msghdl == NULL)
879                 nd_abort(nhdl, "failed to initialize libfmd_msg");

881         (void) gethostname(hostname, MAXHOSTNAMELEN + 1);
882         /*
883          * Set up our event subscriptions.  We subscribe to everything and then
884          * consult libscf when we receive an event to determine whether to send
885          * an email notification.
886          */
887         nd_debug(nhdl, "Subscribing to ireport.* events");
888         if (fmev_shdl_subscribe(nhdl->nh_evhdl, "ireport.*", irpt_cbfunc,
889             NULL) != FMEV_SUCCESS) {
```



```
890                 nd_abort(nhdl, "fmev_shdl_subscribe failed: %s",
891                     fmev_strerror(fmev_errno));
892         }

894         nd_debug(nhdl, "Subscribing to list.* events");
895         if (fmev_shdl_subscribe(nhdl->nh_evhdl, "list.*", listev_cb,
896             NULL) != FMEV_SUCCESS) {
897                 nd_abort(nhdl, "fmev_shdl_subscribe failed: %s",
898                     fmev_strerror(fmev_errno));
899         }

901         /*
902          * We run until someone kills us
903          */
904         while (nhdl->nh_keep_running)
905                 (void) sigsuspend(&set);

907         free(nhdl->nh_rootdir);
908         free(nhdl);

910         return (0);
911 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   20083 Thu Jan 24 09:47:47 2019
new/usr/src/cmd/fm/notify/snmp-notify/common/snmp-notify.c
10130 smatch fixes for usr/src/cmd/fm
**********************************************************
```

```
/*
 * 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) 2010, Oracle and/or its affiliates. All rights reserved.
 */

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

#include <sys/fm/protocol.h>
#include <fm/fmd_snmp.h>
#include <fm/fmd_msg.h>
#include <fm/libfmevent.h>
#include <net-snmp/net-snmp-config.h>
#include <net-snmp/net-snmp-includes.h>
#include <net-snmp/agent/net-snmp-agent-includes.h>
#include <errno.h>
#include <locale.h>
#include <netdb.h>
#include <signal.h>
#include <strings.h>
#include <stdlib.h>
#include <unistd.h>
#include <limits.h>
#include <alloca.h>
#include <priv_utils.h>
#include <zone.h>
#include "libfmnotify.h"

/*
 * Debug messages can be enabled by setting the debug property to true
 *
 * # svccfg -s svc:/system/fm/snmp-notify setprop config/debug=true
 */
#define SVCNAME         "system/fm/snmp-notify"

typedef struct ireport_trap {
        char *host;
        char *msgid;
        char *desc;
        long long tstamp;
```



```
        char *fmri;
        uint32_t from_state;
        uint32_t to_state;
        char *reason;
        boolean_t is_stn_event;
} ireport_trap_t;
______unchanged_portion_omitted_

int
main(int argc, char *argv[])
{
        struct rlimit rlim;
        struct sigaction act;
        sigset_t set;
        char c;
        boolean_t run_fg = B_FALSE;

        if ((nhdl = malloc(sizeof (nd_hdl_t))) == NULL) {
                (void) fprintf(stderr, "Failed to allocate space for notifyd "
                    "handle (%s)", strerror(errno));
                return (1);
        }
        bzero(nhdl, sizeof (nd_hdl_t));
        nhdl->nh_keep_running = B_TRUE;
        nhdl->nh_log_fd = stderr;
        nhdl->nh_pname = argv[0];

        get_svc_config();

        /*
         * In the case where we get started outside of SMF, args passed on the
         * command line override SMF property setting
         */
        while (optind < argc) {
                while ((c = getopt(argc, argv, optstr)) != -1) {
                        switch (c) {
                        case 'd':
                                nhdl->nh_debug = B_TRUE;
                                break;
                        case 'f':
                                run_fg = B_TRUE;
                                break;
                        case 'R':
                                nhdl->nh_rootdir = strdup(optarg);
                                break;
                        default:
                                free(nhdl);
                                return (usage(argv[0]));
                                return (usage(nhdl->nh_pname));
                        }
                }
        }

        /*
         * Set up a signal handler for SIGTERM (and SIGINT if we'll
         * be running in the foreground) to ensure sure we get a chance to exit
         * in an orderly fashion.  We also catch SIGHUP, which will be sent to
         * us by SMF if the service is refreshed.
         */
        (void) sigfillset(&set);
        (void) sigfillset(&act.sa_mask);
        act.sa_handler = nd_sighandler;
        act.sa_flags = 0;

        (void) sigaction(SIGTERM, &act, NULL);
        (void) sigdelset(&set, SIGTERM);
```

```
602         (void) sigaction(SIGHUP, &act, NULL);
603         (void) sigdelset(&set, SIGHUP);

605         if (run_fg) {
606                 (void) sigaction(SIGINT, &act, NULL);
607                 (void) sigdelset(&set, SIGINT);
608         } else
609                 nd_daemonize(nhdl);

611         rlim.rlim_cur = RLIM_INFINITY;
612         rlim.rlim_max = RLIM_INFINITY;
613         (void) setrlimit(RLIMIT_CORE, &rlim);

615         /*
616          * We need to be root initialize our libfmevent handle (because that
617          * involves reading/writing to /dev/sysevent), so we do this before
618          * calling __init_daemon_priv.
619          */
620         nhdl->nh_evhdl = fmev_shdl_init(LIBFMEVENT_VERSION_2, NULL, NULL, NULL);
621         if (nhdl->nh_evhdl == NULL) {
622                 (void) sleep(5);
623                 nd_abort(nhdl, "failed to initialize libfmevent: %s",
624                     fmev_strerror(fmev_errno));
625         }

627         /*
628          * If we're in the global zone, reset all of our privilege sets to
629          * the minimum set of required privileges.   We also change our
630          * uid/gid to noaccess/noaccess
631          *
632          * __init_daemon_priv will also set the process core path for us
633          *
634          */
635         if (getzoneid() == GLOBAL_ZONEID)
636                 if (__init_daemon_priv(
637                     PU_RESETGROUPS | PU_LIMITPRIVS | PU_INHERITPRIVS,
638                     60002, 60002, PRIV_FILE_DAC_READ, NULL) != 0)
639                         nd_abort(nhdl, "additional privileges required to run");

641         nhdl->nh_msghdl = fmd_msg_init(nhdl->nh_rootdir, FMD_MSG_VERSION);
642         if (nhdl->nh_msghdl == NULL)
643                 nd_abort(nhdl, "failed to initialize libfmd_msg");

645         if (init_sma() != SNMPERR_SUCCESS)
646                 nd_abort(nhdl, "SNMP initialization failed");

648         (void) gethostname(hostname, MAXHOSTNAMELEN + 1);
649         /*
650          * Set up our event subscriptions.  We subscribe to everything and then
651          * consult libscf when we receive an event to determine what (if any)
652          * notification to send.
653          */
654         nd_debug(nhdl, "Subscribing to ireport.os.smf.* events");
655         if (fmev_shdl_subscribe(nhdl->nh_evhdl, "ireport.os.smf.*",
656             ireport_cb, NULL) != FMEV_SUCCESS) {
657                 nd_abort(nhdl, "fmev_shdl_subscribe failed: %s",
658                     fmev_strerror(fmev_errno));
659         }

661         nd_debug(nhdl, "Subscribing to list.* events");
662         if (fmev_shdl_subscribe(nhdl->nh_evhdl, "list.*", list_cb,
663             NULL) != FMEV_SUCCESS) {
664                 nd_abort(nhdl, "fmev_shdl_subscribe failed: %s",
665                     fmev_strerror(fmev_errno));
666         }
```



```
668         /*
669          * We run until someone kills us
670          */
671         while (nhdl->nh_keep_running)
672                 (void) sigsuspend(&set);

674         /*
675          * snmp_shutdown, which we would normally use here, calls free_slots,
676          * a callback that is supposed to tear down the pkcs11 state; however,
677          * it abuses C_Finalize, causing fmd to drop core on shutdown.  Avoid
678          * this by shutting down the library piecemeal.
679          */
680         snmp_store(SNMP_SUPPCONF);
681         snmp_alarm_unregister_all();
682         (void) snmp_close_sessions();
683         shutdown_mib();
684         unregister_all_config_handlers();
685         netsnmp_ds_shutdown();

687         free(nhdl->nh_rootdir);
688         free(nhdl);

690         return (0);
691 }
```

_____***unchanged_portion_omitted_***