****
9571 Tue Apr 14 14:09:21 2015
new/exception_lists/check_rtime
3704 libfmd_snmp should compile with newer net-snmp
****

```
#
# 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) 2009, 2010, Oracle and/or its affiliates. All rights reserved.
#

# This file provides exceptions to the usual rules applied to ELF objects by
# check_rtime. All strings are Perl regular expressions that are compared to
# file paths. In addition to the standard Perl syntax, there is one extension:
#
#       MACH(dir)
#
# is expanded into a regular expression that matches the given
# directory, or a 64-bit subdirectory of the directory with the
# name of a 64-bit architecture. For example, MACH(lib) will match
# any of the following:
#
#       lib
#       lib/amd64
#       lib/sparcv9


# Directory hierarchies to skip completely
SKIP            ^usr/lib/libc/                  # optimized libc
SKIP            ^usr/lib/rcm/                   # 4426119
SKIP            ^usr/perl5/                     # alan's taking care of these :-
SKIP            ^usr/src/                       # no objects in source code

# Individual files that we don't examine
SKIP            ^boot/grub/bin/grub$
# USIII specific extns. cause ldd noise on USII bld. m/c
SKIP            ^usr/lib/fps/sun4u/UltraSPARC.*/fptest$
SKIP            ^usr/MACH(lib)/lddstub$         # lddstub has no dependencies
SKIP            ^usr/MACH(lib)/libssagent\.so\.1$       # 4328854
SKIP            ^usr/lib/MACH(iconv)/geniconvtbl.so$    # 4384329

# picl file exclusions (4385799)
SKIP            ^usr/platform/.*/libpsvcplugin_psr\.so\.1
SKIP            ^usr/platform/.*/libpsvcpolicy_psr\.so\.1
SKIP            ^usr/platform/.*/libpsvcpolicy\.so\.1
SKIP            ^usr/lib/sysevent/modules/picl_slm.so$
```



```
# Objects that are allowed to have executable data segments
EXEC_DATA       ^MACH(lib)/ld\.so\.1$
EXEC_DATA       ^lib/libc\.so\.1$       # 6524709, 32-bit, needed for x86 only
EXEC_DATA       ^lib/amd64/libumem\.so\.1$ # ptcumem
EXEC_DATA       ^lib/libumem\.so\.1$    # ptcumem
EXEC_DATA       ^opt/SUNWdtrt/tst/.*/ustack/tst\.helper\.exe$
EXEC_DATA       ^platform/.*/MACH(kernel)/unix$
EXEC_DATA       ^platform/.*/multiboot$

# Objects that are allowed to have an executable stack
EXEC_STACK      ^platform/.*/MACH(kernel)/unix$
EXEC_STACK      ^platform/.*/multiboot$

# Objects for which we allow relocations to the text segment
TEXTREL         ^platform/.*/MACH(kernel)/unix$

# Directories and files that are allowed to have no direct bound symbols
NODIRECT        ^platform/.*/MACH(kernel)/unix$
NODIRECT        ^usr/ucb
NODIRECT        ^usr/4lib/sbcp$

# Identify any files that should be skipped when building a crle(1)
# configuration file.  As the hwcap libraries can be loop-back mounted onto
# libc, these can confuse crle(1) because of their identical dev/inode.
NOCRLEALT       ^usr/lib/libc/libc_hwcap[1-3].so.1$

# Files that should contain debugging information.
STAB    ^platform/.*/MACH(kernel)/unix$

# Files that are allowed undefined references
UNDEF_REF       ^usr/lib/libnisdb\.so\.2$
UNDEF_REF       ^usr/snadm/lib/libsvm\.so\.1$

# Objects allowed to have unused dependencies
UNUSED_DEPS     ^usr/lib/picl/plugins/          # require devtree dependencies
UNUSED_DEPS     ^usr/lib/libp   # profile libc makes libm an unused dep of libc

# libm.so.2 dependency
UNUSED_OBJ      unused object=.*MACH(libm)/libm_hwcap1\.so\.2

# libnetsnmphelpers.so is empty in some net-snmp versions
UNUSED_OBJ      unused object=.*/libnetsnmphelpers\.so\..*
UNREF_OBJ       unreferenced object=.*/libsnmphelpers\.so\..*

# Unused runpaths due to dlopen() use
UNUSED_RPATH    /usr/lib/fs/autofs.*\ from\ .automountd
UNUSED_RPATH    /etc/ppp/plugins.*\ from\ .*pppd
UNUSED_RPATH    /usr/lib/inet/ppp.*\ from\ .*pppd
UNUSED_RPATH    /usr/sfw/lib.*\ from\ .*libipsecutil\.so\.1
UNUSED_RPATH    /usr/platform/.*rsmlib.*\ from\ .*librsm\.so\.2
UNUSED_RPATH    \$ORIGIN.*\ from\ .*fcode.so
UNUSED_RPATH    /opt/VRTSvxvm/lib.*\ from\ .*libdiskmgt\.so\.1

# Unused runpaths in picl code
UNUSED_RPATH    /usr/platform/.*\ from\ .*/usr/platform
UNUSED_RPATH    /usr/lib/picl/.*\ from\ .*/usr/platform
UNUSED_RPATH    /usr/platform/.*\ from\ .*/usr/lib/picl

# Unused runpaths in non-OSNET objects we can't change
UNUSED_RPATH    /usr/lib/mps.*\ from\ .*libnss3\.so
UNUSED_RPATH    /usr/lib/mps.*\ from\ .*libnssutil3\.so
UNUSED_RPATH    /usr/lib/mps.*\ from\ .*libsmime3\.so
UNUSED_RPATH    /usr/lib/mps.*\ from\ .*libssl3\.so
UNUSED_RPATH    /usr/sfw/lib.*\ from\ .*libdbus-1\.so\.3
UNUSED_RPATH    /usr/sfw/lib.*\ from\ .*libdbus-glib-1\.so\.2
UNUSED_RPATH    /usr/sfw/lib.*\ from\ .*libglib-2\.0\.so\.0
```

```
128 UNUSED_RPATH    /usr/X11/lib.*\ from\ .*libglib-2\.0\.so\.0
129 UNUSED_RPATH    /usr/sfw/lib.*\ from\ .*libgobject-2\.0\.so\.0
130 UNUSED_RPATH    /usr/X11/lib.*\ from\ .*libgobject-2\.0\.so\.0
131 UNUSED_RPATH    /usr/sfw/lib.*\ from\ .*libgthread-2\.0\.so\.0
132 UNUSED_RPATH    /usr/X11/lib.*\ from\ .*libgthread-2\.0\.so\.0
133 UNUSED_RPATH    /usr/sfw/lib.*\ from\ .*libcrypto\.so\.0\.9\.8
134 UNUSED_RPATH    /usr/sfw/lib.*\ from\ .*libnetsnmp\.so\..*
130 UNUSED_RPATH    /usr/sfw/lib.*\ from\ .*libnetsnmp\.so\.15
135 UNUSED_RPATH    /usr/sfw/lib.*\ from\ .*libgcc_s\.so\.1
136 UNUSED_RPATH    /usr/ccs/lib.*\ from\ .*libgcc_s\.so\.1
137 UNUSED_RPATH    /usr/lib.*\ from\ .*libgcc_s\.so\.1
138 UNUSED_RPATH    /usr/postgres/8.3/lib.*\ from\ .*libpq\.so\.5
139 UNUSED_RPATH    /usr/sfw/lib.*\ from\ .*libpq\.so\.5
140 UNUSED_RPATH    /usr/lib.*\ from\ .*/usr/lib/mps
141 UNUSED_RPATH    /usr/ccs/lib.*\ from\ .*/usr/lib/mps

143 # Unused runpaths for reasons not captured above
144 UNUSED_RPATH    /usr/lib/smbsrv.*\ from\ .*libsmb\.so\.1        # future needs
145 UNUSED_RPATH    /usr.*\ from\ .*tst\.gcc\.exe                   # gcc built


148 # Unreferenced objects of non-OSnet objects we can't change
149 UNREF_OBJ       /lib.*\ of\ .*libcimapi\.so
150 UNREF_OBJ       /lib.*\ of\ .*libdbus-1\.so\.3
151 UNREF_OBJ       /lib.*\ of\ .*libdbus-glib-1\.so\.2
152 UNREF_OBJ       /lib.*\ of\ .*libglib-2.0\.so\.0
153 UNREF_OBJ       /lib.*\ of\ .*libgobject-2.0\.so\.0
154 UNREF_OBJ       /lib.*\ of\ .*libgthread-2\.0\.so\.0
155 UNREF_OBJ       /lib.*\ of\ .*libjvm\.so
156 UNREF_OBJ       /lib.*\ of\ .*libnetsnmp\.so\..*
157 UNREF_OBJ       /lib.*\ of\ .*libnetsnmpagent\.so\..*
158 UNREF_OBJ       /lib.*\ of\ .*libnetsnmpmibs\.so\..*
159 UNREF_OBJ       /lib.*\ of\ .*libnetsnmphelpers\.so\..*
152 UNREF_OBJ       /lib.*\ of\ .*libnetsnmp\.so\.15
153 UNREF_OBJ       /lib.*\ of\ .*libnetsnmpagent\.so\.15
154 UNREF_OBJ       /lib.*\ of\ .*libnetsnmpmibs\.so\.15
155 UNREF_OBJ       /lib.*\ of\ .*libnetsnmphelpers\.so\.15
160 UNREF_OBJ       /lib.*\ of\ .*libnspr4\.so
161 UNREF_OBJ       /lib.*\ of\ .*libpq\.so\.5
162 UNREF_OBJ       /lib.*\ of\ .*libsoftokn3\.so
163 UNREF_OBJ       /lib.*\ of\ .*libspmicommon\.so\.1
164 UNREF_OBJ       /lib.*\ of\ .*libspmocommon\.so\.1
165 UNREF_OBJ       /lib.*\ of\ .*libssl3\.so
166 UNREF_OBJ       /lib.*\ of\ .*libtspi\.so\.1
167 UNREF_OBJ       /lib.*\ of\ .*libxml2\.so\.2
168 UNREF_OBJ       /lib.*\ of\ .*libxslt\.so\.1
169 UNREF_OBJ       /lib.*\ of\ .*libpq\.so\.4
170 UNREF_OBJ       /lib.*\ of\ .*libpython2\.4\.so\.1\.0
171 UNREF_OBJ       /libgcc_s.*\ of\ .*libstdc\+\+\.so\.6

173 # Unreferenced object of objects we can't change for other reasons
174 UNREF_OBJ       /libmapmalloc\.so\.1;\ unused\ dependency\ of   # interposer
175 UNREF_OBJ       /libstdc\+\+\.so\.6;\ unused\ dependency\ of    # gcc build
176 UNREF_OBJ       /libm\.so\.2.*\ of\ .*libstdc\+\+\.so\.6        # gcc build
177 UNREF_OBJ       /lib.*\ of\ .*/lib/picl/plugins/                # picl
178 UNREF_OBJ       /lib.*\ of\ .*kcfd                              # interposer
179 UNREF_OBJ       /libpkcs11\.so\.1; .*\ of\ .*libkmf\.so\.1      # interposed
180 # Referenced by the Studio build, not the GCC build.  GCC eliminates the unused
181 # statics which have the dependence.
182 UNREF_OBJ       /libc\.so\.1.*\ of\ .*kldap\.so\.1


185 # Objects that used to contain system functionalty that has since
186 # migrated to libc. We preserve these libraries as pure filters for
187 # backward compatability but nothing needs to link to them.
188 OLDDEP          libaio\.so\.1                   # onnv build 44
```



```
189 OLDDEP          libdl\.so\.1                    # on10 build 49
190 OLDDEP          libdoor\.so\.1                  # onnv build 12
191 OLDDEP          libintl\.so\.1                  # on297 build 7
192 OLDDEP          libpthread\.so\.1               # on10 build 53
193 OLDDEP          librt\.so\.1                    # onnv build 44
194 OLDDEP          libsched\.so\.1                 # on10 build 36
195 OLDDEP          libthread\.so\.1                # on10 build 53
196 OLDDEP          libw\.so\.1                     # on297 build 7

198 # Files for which we skip checking of duplicate addresses in the
199 # symbol sort sections. Such exceptions should be rare --- most code will
200 # not have duplicate addresses, since it takes assember or a "#pragma weak"
201 # to do such aliasing in C. C++ is different: The compiler generates aliases
202 # for implementation reasons, and the mangled names used to encode argument
203 # and return value types are difficult to handle well in mapfiles.
204 # Furthermore, the Sun compiler and gcc use different and incompatible
205 # name mangling conventions. Since ON must be buildable by either, we
206 # would have to maintain two sets of mapfiles for each such object.
207 # C++ use is rare in ON, so this is not worth pursuing.
208 #
209 NOSYMSORT       opt/SUNWdtrt/tst/common/pid/tst.weak2.exe       # DTrace test
210 NOSYMSORT       lib/amd64/libnsl\.so\.1                         # C++
211 NOSYMSORT       lib/sparcv9/libnsl\.so\.1                       # C++
212 NOSYMSORT       lib/sparcv9/libfru\.so\.1                       # C++
213 NOSYMSORT       usr/lib/lms                                     # C++
214 NOSYMSORT       ld\.so\.1                                       # libc_pic.a use
215 NOSYMSORT       lib/libsun_fc\.so\.1                            # C++
216 NOSYMSORT       lib/amd64/libsun_fc\.so\.1                      # C++
217 NOSYMSORT       lib/sparcv9/libsun_fc\.so\.1                    # C++
218 NOSYMSORT       usr/lib/amd64/libfru\.so\.1                     # C++


221 # The libprtdiag_psr.so.1 objects built under usr/src/lib/libprtdiag_psr
222 # are a family, all built using the same makefile, targeted at different
223 # sparc hardware variants. There are a small number of cases where this
224 # one size fits all approach causes an object to be linked against an
225 # unneeded library.
226 UNREF_OBJ       lib/(libdevinfo|libcfgadm)\.so\.1; .*\ of\ .*SUNW,Netra-CP2300/l
```

```
**********************************************************
    2124 Tue Apr 14 14:09:21 2015
new/usr/src/lib/fm/libfmd_snmp/Makefile.com
3704 libfmd_snmp should compile with newer net-snmp
**********************************************************
```

```
#
# 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 2008 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

LIBRARY = libfmd_snmp.a
VERS = .1

LIBSRCS = \
        debug_subr.c    \
        init.c          \
        module.c        \
        problem.c       \
        resource.c      \
        scheme.c

OBJECTS = $(LIBSRCS:%.c=%.o)

include ../../../Makefile.lib
include ../../Makefile.lib

SRCS = $(LIBSRCS:%.c=../common/%.c)
LIBS = $(DYNLIB) $(LINTLIB)

SRCDIR =        ../common

C99MODE= $(C99_ENABLE)

CPPFLAGS += -I../common -I.
$(NOT_RELEASE_BUILD)CPPFLAGS += -DDEBUG
CFLAGS += $(CCVERBOSE) $(C_BIGPICFLAGS)
CFLAGS64 += $(CCVERBOSE) $(C_BIGPICFLAGS)

# No lint libraries are delivered for Net-SNMP yet
SNMPLIBS = -lnetsnmp -lnetsnmphelpers -lnetsnmpagent
lint := SNMPLIBS=

LDLIBS += $(MACH_LDLIBS)
LDLIBS += -lfmd_adm -luutil -lnvpair -ltopo
LDLIBS += $(SNMPLIBS)
LDLIBS += -lc
```



```
LINTFLAGS = -msux $(C99LMODE)
LINTFLAGS64 = -msux -m64 $(C99LMODE)
LINTFLAGS = -msux
LINTFLAGS64 = -msux -m64

# Net-SNMP's headers use do {} while (0) a lot
LINTCHECKFLAGS += -erroff=E_CONSTANT_CONDITION

$(LINTLIB) := SRCS = $(SRCDIR)/$(LINTSRC)
$(LINTLIB) := LINTFLAGS = -nsvx
$(LINTLIB) := LINTFLAGS64 = -nsvx -m64

.KEEP_STATE:

all: $(LIBS)

lint: $(LINTLIB) lintcheck

pics/%.o: ../$(MACH)/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

%.o: ../common/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

include ../../../Makefile.targ
include ../../Makefile.targ
```

****
21161 Tue Apr 14 14:09:21 2015
new/usr/src/lib/fm/libfmd_snmp/common/module.c
5835 fix printf tokens for net-snmp 5.7.2
****
_____***unchanged_portion_omitted_***

```
161 /*
162  * Possible update the contents of a single module within the cache.  This
163  * is our callback from fmd_module_iter.
164  */
165 static int
166 modinfo_update_one(const fmd_adm_modinfo_t *modinfo, void *arg)
167 {
168         const sunFmModule_update_ctx_t *update_ctx =
169             (sunFmModule_update_ctx_t *)arg;
170         sunFmModule_data_t *data = module_lookup_name(modinfo->ami_name);

172         /*
173          * An fmd module we haven't seen before.  We're obligated to index
174          * it and link it into our cache so that we can find it, but we're
175          * not obligated to fill it in completely unless we're doing a
176          * thorough update or this is the module we were asked for.  This
177          * avoids unnecessary iteration and memory manipulation for data
178          * we're not going to return for this request.
179          */
180         if (data == NULL) {
181                 uu_avl_index_t idx;

183                 DEBUGMSGTL((MODNAME_STR, "found new fmd module %s\n",
184                     modinfo->ami_name));
185                 if ((data = SNMP_MALLOC_TYPEDEF(sunFmModule_data_t)) == NULL) {
186                         (void) snmp_log(LOG_ERR, MODNAME_STR ": Out of memory "
187                             "for new module data at %s:%d\n", __FILE__,
188                             __LINE__);
189                         return (1);
190                 }
191                 /*
192                  * We allocate indices sequentially and never reuse them.
193                  * This ensures we can always return valid GETNEXT responses
194                  * without having to reindex, and it provides the user a
195                  * more consistent view of the fault manager.
196                  */
197                 data->d_index = ++max_index;
198                 DEBUGMSGTL((MODNAME_STR, "index %lu is %s@%p\n", data->d_index,
199                     modinfo->ami_name, data));

201                 (void) strlcpy(data->d_ami_name, modinfo->ami_name,
202                     sizeof (data->d_ami_name));

204                 uu_avl_node_init(data, &data->d_name_avl, mod_name_avl_pool);
205                 (void) uu_avl_find(mod_name_avl, data, NULL, &idx);
206                 uu_avl_insert(mod_name_avl, data, idx);

208                 uu_avl_node_init(data, &data->d_index_avl, mod_index_avl_pool);
209                 (void) uu_avl_find(mod_index_avl, data, NULL, &idx);
210                 uu_avl_insert(mod_index_avl, data, idx);

212                 DEBUGMSGTL((MODNAME_STR, "completed new module %lu/%s@%p\n",
213                     data->d_index, data->d_ami_name, data));
214         }

216         data->d_valid = valid_stamp;

218         DEBUGMSGTL((MODNAME_STR, "timestamp updated for %lu/%s@%p: %d\n",
218         DEBUGMSGTL((MODNAME_STR, "timestamp updated for %lu/%s@%p: %lu\n",
```



```
219             data->d_index, data->d_ami_name, data, data->d_valid));

221         if ((update_ctx->uc_type & UCT_ALL) ||
222             update_ctx->uc_index == data->d_index) {
223                 (void) strlcpy(data->d_ami_vers, modinfo->ami_vers,
224                     sizeof (data->d_ami_vers));
225                 (void) strlcpy(data->d_ami_desc, modinfo->ami_desc,
226                     sizeof (data->d_ami_desc));
227                 data->d_ami_flags = modinfo->ami_flags;
228         }

230         return (!(update_ctx->uc_type & UCT_ALL) &&
231             update_ctx->uc_index == data->d_index);
232 }
```

_____***unchanged_portion_omitted_***

****
30965 Tue Apr 14 14:09:21 2015
new/usr/src/lib/fm/libfmd_snmp/common/problem.c
5835 fix printf tokens for net-snmp 5.7.2
****
_____***unchanged_portion_omitted_***

```
 468 /*
 469  * Returns the problem data for the problem whose uuid is next according
 470  * to ASN.1 lexical ordering after the request in table_info.  Indexes are
 471  * updated to reflect the OID of the value being returned.  This allows
 472  * us to implement GETNEXT.
 473  */
 474 static sunFmProblem_data_t *
 475 sunFmProblemTable_nextpr(netsnmp_handler_registration *reginfo,
 476     netsnmp_table_request_info *table_info)
 477 {
 478         sunFmProblem_data_t     *data;
 479         char                    *uuid = "";

 481         if (table_info->number_indexes < 1) {
 482                 oid tmpoid[MAX_OID_LEN];

 484                 DEBUGMSGTL((MODNAME_STR, "nextpr: no indexes given\n"));

 486                 snmp_free_varbind(table_info->indexes);
 487                 table_info->indexes =
 488                     SNMP_MALLOC_TYPEDEF(netsnmp_variable_list);
 489                 (void) snmp_set_var_typed_value(table_info->indexes,
 490                     ASN_OCTET_STR, (const uchar_t *)uuid, 0);
 491                 (void) memcpy(tmpoid, reginfo->rootoid,
 492                     reginfo->rootoid_len * sizeof (oid));
 493                 tmpoid[reginfo->rootoid_len] = 1;
 494                 tmpoid[reginfo->rootoid_len + 1] = table_info->colnum;
 495                 if (build_oid_segment(table_info->indexes) != SNMPERR_SUCCESS) {
 496                         snmp_free_varbind(table_info->indexes);
 497                         return (NULL);
 498                 }
 499                 table_info->number_indexes = 1;
 500                 table_info->index_oid_len = table_info->indexes->name_length;
 501                 (void) memcpy(table_info->index_oid, table_info->indexes->name,
 502                     table_info->indexes->name_length);

 504                 DEBUGMSGTL((MODNAME_STR, "nextpr: built fake index:\n"));
 505                 DEBUGMSGVAR((MODNAME_STR, table_info->indexes));
 506                 DEBUGMSG((MODNAME_STR, "\n"));
 507         } else {
 508                 /*
 509                  * Construct the next possible UUID to look for.  We can
 510                  * simply increment the least significant byte of the last
 511                  * UUID because (a) that preserves SNMP lex order and (b)
 512                  * the characters that may appear in a UUID do not include
 513                  * 127 nor 255.
 514                  */
 515                 uuid = alloca(table_info->indexes->val_len + 1);
 516                 (void) strlcpy(uuid,
 517                     (const char *)table_info->indexes->val.string,
 518                     table_info->indexes->val_len + 1);
 519                 ++uuid[table_info->indexes->val_len - 1];

 521                 DEBUGMSGTL((MODNAME_STR, "nextpr: received index:\n"));
 522                 DEBUGMSGVAR((MODNAME_STR, table_info->indexes));
 523                 DEBUGMSG((MODNAME_STR, "\n"));
 524         }

 526         if ((data = problem_lookup_uuid_next(uuid)) == NULL) {
```



```
 527                 DEBUGMSGTL((MODNAME_STR, "nextpr: next match not found for "
 528                     "%s; trying next column\n", uuid));
 529                 if (table_info->colnum >=
 530                     netsnmp_find_table_registration_info(reginfo)->max_column) {
 531                         snmp_free_varbind(table_info->indexes);
 532                         table_info->indexes = NULL;
 533                         table_info->number_indexes = 0;
 534                         DEBUGMSGTL((MODNAME_STR, "nextpr: out of columns\n"));
 535                         return (NULL);
 536                 }
 537                 table_info->colnum++;
 538                 DEBUGMSGTL((MODNAME_STR, "nextpr: search for col %u empty "
 539                     "uuid\n", table_info->colnum));
 539                     "uuid\n", table_info->colnum, uuid));

 541                 if ((data = problem_lookup_uuid_next("")) == NULL) {
 542                         DEBUGMSGTL((MODNAME_STR, "nextpr: next match not found "
 543                             "for empty uuid; stopping\n"));
 544                         snmp_free_varbind(table_info->indexes);
 545                         table_info->indexes = NULL;
 546                         table_info->number_indexes = 0;
 547                         return (NULL);
 548                 }
 549         }

 551         (void) snmp_set_var_typed_value(table_info->indexes, ASN_OCTET_STR,
 552             (uchar_t *)data->d_aci_uuid, strlen(data->d_aci_uuid));
 553         table_info->number_indexes = 1;

 555         DEBUGMSGTL((MODNAME_STR, "matching data is %s@%p\n", data->d_aci_uuid,
 556             data));

 558         return (data);
 559 }
```

_____***unchanged_portion_omitted_***

****
23872 Tue Apr 14 14:09:22 2015
new/usr/src/lib/fm/libfmd_snmp/common/resource.c
5835 fix printf tokens for net-snmp 5.7.2
****
_____***unchanged_portion_omitted_***

```
145 /*
146  * Possible update the contents of a single resource within the cache.  This
147  * is our callback from fmd_rsrc_iter.
148  */
149 static int
150 rsrcinfo_update_one(const fmd_adm_rsrcinfo_t *rsrcinfo, void *arg)
151 {
152         const sunFmResource_update_ctx_t *update_ctx =
153             (sunFmResource_update_ctx_t *)arg;
154         sunFmResource_data_t *data = resource_lookup_fmri(rsrcinfo->ari_fmri);

156         ++rsrc_count;

158         /*
159          * A resource we haven't seen before.  We're obligated to index
160          * it and link it into our cache so that we can find it, but we're
161          * not obligated to fill it in completely unless we're doing a
162          * full update or this is the resource we were asked for.  This
163          * avoids unnecessary iteration and memory manipulation for data
164          * we're not going to return for this request.
165          */
166         if (data == NULL) {
167                 uu_avl_index_t idx;

169                 DEBUGMSGTL((MODNAME_STR, "found new resource %s\n",
170                     rsrcinfo->ari_fmri));
171                 if ((data = SNMP_MALLOC_TYPEDEF(sunFmResource_data_t)) ==
172                     NULL) {
173                         (void) snmp_log(LOG_ERR, MODNAME_STR ": Out of memory "
174                             "for new resource data at %s:%d\n", __FILE__,
175                             __LINE__);
176                         return (1);
177                 }
178                 /*
179                  * We allocate indices sequentially and never reuse them.
180                  * This ensures we can always return valid GETNEXT responses
181                  * without having to reindex, and it provides the user a
182                  * more consistent view of the fault manager.
183                  */
184                 data->d_index = ++max_index;
185                 DEBUGMSGTL((MODNAME_STR, "index %lu is %s@%p\n", data->d_index,
186                     rsrcinfo->ari_fmri, data));

188                 (void) strlcpy(data->d_ari_fmri, rsrcinfo->ari_fmri,
189                     sizeof (data->d_ari_fmri));

191                 uu_avl_node_init(data, &data->d_fmri_avl, rsrc_fmri_avl_pool);
192                 (void) uu_avl_find(rsrc_fmri_avl, data, NULL, &idx);
193                 uu_avl_insert(rsrc_fmri_avl, data, idx);

195                 uu_avl_node_init(data, &data->d_index_avl, rsrc_index_avl_pool);
196                 (void) uu_avl_find(rsrc_index_avl, data, NULL, &idx);
197                 uu_avl_insert(rsrc_index_avl, data, idx);

199                 DEBUGMSGTL((MODNAME_STR, "completed new resource %lu/%s@%p\n",
200                     data->d_index, data->d_ari_fmri, data));
201         }

203         data->d_valid = valid_stamp;
```



```
205         DEBUGMSGTL((MODNAME_STR, "timestamp updated for %lu/%s@%p: %d\n",
205         DEBUGMSGTL((MODNAME_STR, "timestamp updated for %lu/%s@%p: %lu\n",
206             data->d_index, data->d_ari_fmri, data, data->d_valid));

208         if ((update_ctx->uc_type & UCT_ALL) ||
209             update_ctx->uc_index == data->d_index) {
210                 (void) strlcpy(data->d_ari_case, rsrcinfo->ari_case,
211                     sizeof (data->d_ari_case));
212                 data->d_ari_flags = rsrcinfo->ari_flags;
213         }

215         return (!(update_ctx->uc_type & UCT_ALL) &&
216             update_ctx->uc_index == data->d_index);
217 }
```

_____***unchanged_portion_omitted_***