```
**********************************************************
   57841 Fri Jan 12 10:52:06 2018
new/usr/src/lib/fm/topo/libtopo/common/topo_xml.c
8954 libtopo cannot handle any array type other than string_array.
Reviewed by: Andy Stormont astormont@racktopsystems.com
Reviewed by: David H ˆ¶ppner 0xffea@gmail.com
Reviewed by: Rob Johnston rob.johnston@joyent.com
**********************************************************
```

```
/*
 * 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) 2013, Joyent, Inc. All rights reserved.
 * Copyright (c) 2018, Western Digital Technologies, Inc. All rights reserved.
 */

#include <libxml/parser.h>
#include <libxml/xinclude.h>
#include <sys/fm/protocol.h>
#include <assert.h>
#include <string.h>
#include <strings.h>
#include <ctype.h>
#include <errno.h>
#include <limits.h>
#include <fm/libtopo.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <topo_file.h>
#include <topo_mod.h>
#include <topo_subr.h>
#include <topo_alloc.h>
#include <topo_parse.h>
#include <topo_error.h>

static tf_rdata_t *topo_xml_walk(topo_mod_t *, tf_info_t *, xmlNodePtr,
    tnode_t *);
static tf_edata_t *enum_attributes_process(topo_mod_t *, xmlNodePtr);
static int enum_run(topo_mod_t *, tf_rdata_t *);
static int fac_enum_run(topo_mod_t *, tnode_t *, const char *);
static int fac_process(topo_mod_t *, xmlNodePtr, tf_rdata_t *, tnode_t *);
static int fac_enum_process(topo_mod_t *, xmlNodePtr, tnode_t *);
static int decorate_nodes(topo_mod_t *, tf_rdata_t *, xmlNodePtr, tnode_t *,
    tf_pad_t **);
```



```
static void
strarr_free(topo_mod_t *mod, char **arr, uint_t nelems)
{
        int i;

        for (i = 0; i < nelems; i++)
                topo_mod_strfree(mod, arr[i]);
        topo_mod_free(mod, arr, (nelems * sizeof (char *)));
}
```

_____*unchanged_portion_omitted_*

```
static int
xlate_common(topo_mod_t *mp, xmlNodePtr xn, topo_type_t ptype, nvlist_t *nvl,
    const char *name)
const char *name)
{
        int rv;
        uint64_t ui;
        uint_t i = 0, nelems = 0;
        nvlist_t *fmri;
        xmlChar *str;
        char **strarrbuf;
        void *arrbuf;
        nvlist_t **nvlarrbuf;
        xmlNodePtr cn;

        topo_dprintf(mp->tm_hdl, TOPO_DBG_XML, "xlate_common(name=%s)\n", name);
        switch (ptype) {
        case TOPO_TYPE_INT32:
                if (xmlattr_to_int(mp, xn, Value, &ui) < 0)
                        return (-1);
                rv = nvlist_add_int32(nvl, name, (int32_t)ui);
                break;
        case TOPO_TYPE_UINT32:
                if (xmlattr_to_int(mp, xn, Value, &ui) < 0)
                        return (-1);
                rv = nvlist_add_uint32(nvl, name, (uint32_t)ui);
                break;
        case TOPO_TYPE_INT64:
                if (xmlattr_to_int(mp, xn, Value, &ui) < 0)
                        return (-1);
                rv = nvlist_add_int64(nvl, name, (int64_t)ui);
                break;
        case TOPO_TYPE_UINT64:
                if (xmlattr_to_int(mp, xn, Value, &ui) < 0)
                        return (-1);
                rv = nvlist_add_uint64(nvl, name, ui);
                break;
        case TOPO_TYPE_FMRI:
                if (xmlattr_to_fmri(mp, xn, Value, &fmri) < 0)
                        return (-1);
                rv = nvlist_add_nvlist(nvl, name, fmri);
                nvlist_free(fmri);
                break;
        case TOPO_TYPE_STRING:
                if ((str = xmlGetProp(xn, (xmlChar *)Value)) == NULL)
                        return (-1);
                rv = nvlist_add_string(nvl, name, (char *)str);
                xmlFree(str);
                break;
        case TOPO_TYPE_INT32_ARRAY:
        case TOPO_TYPE_UINT32_ARRAY:
        case TOPO_TYPE_INT64_ARRAY:
        case TOPO_TYPE_UINT64_ARRAY:
                for (cn = xn->xmlChildrenNode; cn != NULL; cn = cn->next)
                        if ((xmlStrcmp(cn->name, (xmlChar *)Propitem) == 0) ||
```

```
                            (xmlStrcmp(cn->name, (xmlChar *)Argitem) == 0))
                                nelems++;

                if (nelems < 1) {
                        topo_dprintf(mp->tm_hdl, TOPO_DBG_ERR, "No <propitem> "
                            "or <argitem> elements found for array val");
                        return (-1);
                }
                if ((arrbuf = topo_mod_alloc(mp, (nelems * sizeof (uint64_t))))
                    == NULL)
                        return (topo_mod_seterrno(mp, ETOPO_NOMEM));
                break;
        case TOPO_TYPE_STRING_ARRAY:
        case TOPO_TYPE_FMRI_ARRAY:
                for (cn = xn->xmlChildrenNode; cn != NULL; cn = cn->next)
                        if ((xmlStrcmp(cn->name, (xmlChar *)Propitem) == 0) ||
                            (xmlStrcmp(cn->name, (xmlChar *)Argitem) == 0))
                                nelems++;

                if (nelems < 1) {
                        topo_dprintf(mp->tm_hdl, TOPO_DBG_ERR, "No <propitem> "
                            "or <argitem> elements found for array val");
                        return (-1);
                }
                if ((strarrbuf = topo_mod_alloc(mp, (nelems * sizeof (char *))))
                    == NULL)
                        return (topo_mod_seterrno(mp, ETOPO_NOMEM));
                break;
        case TOPO_TYPE_FMRI_ARRAY:
                for (cn = xn->xmlChildrenNode; cn != NULL; cn = cn->next)
                        if ((xmlStrcmp(cn->name, (xmlChar *)Propitem) == 0) ||
                            (xmlStrcmp(cn->name, (xmlChar *)Argitem) == 0))
                                nelems++;

                if (nelems < 1) {
                        topo_dprintf(mp->tm_hdl, TOPO_DBG_ERR, "No <propitem> "
                            "elements found for array prop");
                        return (-1);
                }
                if ((nvlarrbuf = topo_mod_alloc(mp, (nelems *
                    sizeof (nvlist_t *)))) == NULL)
                        return (topo_mod_seterrno(mp, ETOPO_NOMEM));
                break;
        default:
                topo_dprintf(mp->tm_hdl, TOPO_DBG_ERR,
                    "Unrecognized type attribute (ptype = %d)\n", ptype);
                return (topo_mod_seterrno(mp, ETOPO_PRSR_BADTYPE));
        }

        switch (ptype) {
        case TOPO_TYPE_INT32_ARRAY:
                if ((arrbuf = topo_mod_alloc(mp, (nelems * sizeof (int32_t))))
                    == NULL)
                        return (topo_mod_seterrno(mp, ETOPO_NOMEM));
                for (cn = xn->xmlChildrenNode; cn != NULL; cn = cn->next) {
                        if ((xmlStrcmp(cn->name, (xmlChar *)Propitem) == 0) ||
                            (xmlStrcmp(cn->name, (xmlChar *)Argitem) == 0)) {

                                if ((str = xmlGetProp(cn, (xmlChar *)Value))
                                if ((str = xmlGetProp(xn, (xmlChar *)Value))
                                    == NULL)
                                        return (-1);

                                ((int32_t *)arrbuf)[i++]
                                    = atoi((const char *)str);
                                xmlFree(str);
```



```
                        }
                }

                rv = nvlist_add_int32_array(nvl, name, (int32_t *)arrbuf,
                    nelems);
                topo_mod_free(mp, arrbuf, (nelems * sizeof (int32_t)));
                free(arrbuf);
                break;
        case TOPO_TYPE_UINT32_ARRAY:
                if ((arrbuf = topo_mod_alloc(mp, (nelems * sizeof (uint32_t))))
                    == NULL)
                        return (topo_mod_seterrno(mp, ETOPO_NOMEM));
                for (cn = xn->xmlChildrenNode; cn != NULL; cn = cn->next) {
                        if ((xmlStrcmp(cn->name, (xmlChar *)Propitem) == 0) ||
                            (xmlStrcmp(cn->name, (xmlChar *)Argitem) == 0)) {

                                if ((str = xmlGetProp(cn, (xmlChar *)Value))
                                if ((str = xmlGetProp(xn, (xmlChar *)Value))
                                    == NULL)
                                        return (-1);

                                ((uint32_t *)arrbuf)[i++]
                                    = atoi((const char *)str);
                                xmlFree(str);
                        }
                }

                rv = nvlist_add_uint32_array(nvl, name, (uint32_t *)arrbuf,
                    nelems);
                topo_mod_free(mp, arrbuf, (nelems * sizeof (uint32_t)));
                free(arrbuf);
                break;
        case TOPO_TYPE_INT64_ARRAY:
                if ((arrbuf = topo_mod_alloc(mp, (nelems * sizeof (int64_t))))
                    == NULL)
                        return (topo_mod_seterrno(mp, ETOPO_NOMEM));
                for (cn = xn->xmlChildrenNode; cn != NULL; cn = cn->next) {
                        if ((xmlStrcmp(cn->name, (xmlChar *)Propitem) == 0) ||
                            (xmlStrcmp(cn->name, (xmlChar *)Argitem) == 0)) {

                                if ((str = xmlGetProp(cn, (xmlChar *)Value))
                                if ((str = xmlGetProp(xn, (xmlChar *)Value))
                                    == NULL)
                                        return (-1);

                                ((int64_t *)arrbuf)[i++]
                                    = atol((const char *)str);
                                xmlFree(str);
                        }
                }

                rv = nvlist_add_int64_array(nvl, name, (int64_t *)arrbuf,
                    nelems);
                topo_mod_free(mp, arrbuf, (nelems * sizeof (int64_t)));
                free(arrbuf);
                break;
        case TOPO_TYPE_UINT64_ARRAY:
                if ((arrbuf = topo_mod_alloc(mp, (nelems * sizeof (uint64_t))))
                    == NULL)
                        return (topo_mod_seterrno(mp, ETOPO_NOMEM));
                for (cn = xn->xmlChildrenNode; cn != NULL; cn = cn->next) {
                        if ((xmlStrcmp(cn->name, (xmlChar *)Propitem) == 0) ||
                            (xmlStrcmp(cn->name, (xmlChar *)Argitem) == 0)) {

                                if ((str = xmlGetProp(cn, (xmlChar *)Value))
                                if ((str = xmlGetProp(xn, (xmlChar *)Value))
```

```
343                                     == NULL)
344                                         return (-1);
345
346                                 ((uint64_t *)arrbuf)[i++]
347                                     = atol((const char *)str);
348                                 xmlFree(str);
349                         }
350                 }
351
352                 rv = nvlist_add_uint64_array(nvl, name, arrbuf,
353                     nelems);
354                 topo_mod_free(mp, arrbuf, (nelems * sizeof (uint64_t)));
372                 free(arrbuf);
355                 break;
356         case TOPO_TYPE_STRING_ARRAY:
357                 if ((strarrbuf = topo_mod_alloc(mp, (nelems * sizeof (char *))))
358                     == NULL)
359                         return (topo_mod_seterrno(mp, ETOPO_NOMEM));
360                 for (cn = xn->xmlChildrenNode; cn != NULL; cn = cn->next) {
361                         if ((xmlStrcmp(cn->name, (xmlChar *)Propitem) == 0) ||
362                             (xmlStrcmp(cn->name, (xmlChar *)Argitem) == 0)) {
363
364                                 if ((str = xmlGetProp(cn, (xmlChar *)Value))
365                                     == NULL)
366                                         return (-1);
367
368                                 strarrbuf[i++] =
369                                     topo_mod_strdup(mp, (const char *)str);
370                                 xmlFree(str);
371                         }
372                 }
373
374                 rv = nvlist_add_string_array(nvl, name, strarrbuf, nelems);
375                 strarr_free(mp, strarrbuf, nelems);
376                 break;
377         case TOPO_TYPE_FMRI_ARRAY:
378                 if ((nvlarrbuf = topo_mod_alloc(mp, (nelems *
379                     sizeof (nvlist_t *)))) == NULL)
380                         return (topo_mod_seterrno(mp, ETOPO_NOMEM));
381                 for (cn = xn->xmlChildrenNode; cn != NULL; cn = cn->next) {
382                         if ((xmlStrcmp(cn->name, (xmlChar *)Propitem) == 0) ||
383                             (xmlStrcmp(cn->name, (xmlChar *)Argitem) == 0)) {
384
385                                 if ((str = xmlGetProp(cn, (xmlChar *)Value))
397                                 if ((str = xmlGetProp(xn, (xmlChar *)Value))
386                                     == NULL)
387                                         return (-1);
388
389                                 if (topo_mod_str2nvl(mp, (const char *)str,
390                                     &(nvlarrbuf[i++])) < 0) {
391                                         xmlFree(str);
392                                         return (-1);
393                                 }
394                                 xmlFree(str);
395                         }
396                 }
397
398                 rv = nvlist_add_nvlist_array(nvl, name, nvlarrbuf,
399                     nelems);
400                 topo_mod_free(mp, nvlarrbuf, (nelems * sizeof (nvlist_t *)));
412                 free(nvlarrbuf);
401                 break;
402         }
403
404         if (rv != 0) {
405                 topo_dprintf(mp->tm_hdl, TOPO_DBG_ERR,
```



```
406                     "Nvlist construction failed.\n");
407                 return (topo_mod_seterrno(mp, ETOPO_NOMEM));
408         } else
409                 return (0);
410 }
```

_____unchanged_portion_omitted_