```
**********************************************************
   27319 Sat Sep 14 03:53:20 2013
new/usr/src/lib/libbrand/common/libbrand.c
2249 libbrand suppresses libxml2 errors
Reviewed by: Garrett D'Amore <garrett@damore.org>
Reviewed by: Eric Shrock <eric.schrock@delphix.com>
Reviewed by: Milan Jurik <milan.jurik@xylab.cz>
**********************************************************
```

```
/*
 * 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) 2012 Nexenta Systems, Inc.  All rights reserved.
#endif /* ! codereview */
 */

#include <assert.h>
#include <dirent.h>
#include <errno.h>
#include <fnmatch.h>
#include <signal.h>
#include <stdlib.h>
#include <unistd.h>
#include <strings.h>
#include <synch.h>
#include <sys/brand.h>
#include <sys/fcntl.h>
#include <sys/param.h>
#include <sys/stat.h>
#include <sys/systeminfo.h>
#include <sys/types.h>
#include <thread.h>
#include <zone.h>

#include <libbrand_impl.h>
#include <libbrand.h>

#define DTD_ELEM_ATTACH         ((const xmlChar *) "attach")
#define DTD_ELEM_BOOT           ((const xmlChar *) "boot")
#define DTD_ELEM_BRAND          ((const xmlChar *) "brand")
#define DTD_ELEM_CLONE          ((const xmlChar *) "clone")
#define DTD_ELEM_COMMENT        ((const xmlChar *) "comment")
#define DTD_ELEM_DETACH         ((const xmlChar *) "detach")
#define DTD_ELEM_DEVICE         ((const xmlChar *) "device")
#define DTD_ELEM_GLOBAL_MOUNT   ((const xmlChar *) "global_mount")
#define DTD_ELEM_HALT           ((const xmlChar *) "halt")
#define DTD_ELEM_INITNAME       ((const xmlChar *) "initname")
```



```
#define DTD_ELEM_INSTALL        ((const xmlChar *) "install")
#define DTD_ELEM_INSTALLOPTS    ((const xmlChar *) "installopts")
#define DTD_ELEM_LOGIN_CMD      ((const xmlChar *) "login_cmd")
#define DTD_ELEM_FORCELOGIN_CMD ((const xmlChar *) "forcedlogin_cmd")
#define DTD_ELEM_MODNAME        ((const xmlChar *) "modname")
#define DTD_ELEM_MOUNT          ((const xmlChar *) "mount")
#define DTD_ELEM_POSTATTACH     ((const xmlChar *) "postattach")
#define DTD_ELEM_POSTCLONE      ((const xmlChar *) "postclone")
#define DTD_ELEM_POSTINSTALL    ((const xmlChar *) "postinstall")
#define DTD_ELEM_POSTSNAP       ((const xmlChar *) "postsnap")
#define DTD_ELEM_POSTSTATECHG   ((const xmlChar *) "poststatechange")
#define DTD_ELEM_PREDETACH      ((const xmlChar *) "predetach")
#define DTD_ELEM_PRESNAP        ((const xmlChar *) "presnap")
#define DTD_ELEM_PRESTATECHG    ((const xmlChar *) "prestatechange")
#define DTD_ELEM_PREUNINSTALL   ((const xmlChar *) "preuninstall")
#define DTD_ELEM_PRIVILEGE      ((const xmlChar *) "privilege")
#define DTD_ELEM_QUERY          ((const xmlChar *) "query")
#define DTD_ELEM_SYMLINK        ((const xmlChar *) "symlink")
#define DTD_ELEM_SYSBOOT        ((const xmlChar *) "sysboot")
#define DTD_ELEM_UNINSTALL      ((const xmlChar *) "uninstall")
#define DTD_ELEM_USER_CMD       ((const xmlChar *) "user_cmd")
#define DTD_ELEM_VALIDSNAP      ((const xmlChar *) "validatesnap")
#define DTD_ELEM_VERIFY_CFG     ((const xmlChar *) "verify_cfg")
#define DTD_ELEM_VERIFY_ADM     ((const xmlChar *) "verify_adm")

#define DTD_ATTR_ALLOWEXCL      ((const xmlChar *) "allow-exclusive-ip")
#define DTD_ATTR_ARCH           ((const xmlChar *) "arch")
#define DTD_ATTR_DIRECTORY      ((const xmlChar *) "directory")
#define DTD_ATTR_IPTYPE         ((const xmlChar *) "ip-type")
#define DTD_ATTR_MATCH          ((const xmlChar *) "match")
#define DTD_ATTR_MODE           ((const xmlChar *) "mode")
#define DTD_ATTR_NAME           ((const xmlChar *) "name")
#define DTD_ATTR_OPT            ((const xmlChar *) "opt")
#define DTD_ATTR_PATH           ((const xmlChar *) "path")
#define DTD_ATTR_SET            ((const xmlChar *) "set")
#define DTD_ATTR_SOURCE         ((const xmlChar *) "source")
#define DTD_ATTR_SPECIAL        ((const xmlChar *) "special")
#define DTD_ATTR_TARGET         ((const xmlChar *) "target")
#define DTD_ATTR_TYPE           ((const xmlChar *) "type")

#define DTD_ENTITY_TRUE         "true"

static volatile boolean_t       libbrand_initialized = B_FALSE;
static char                     i_curr_arch[MAXNAMELEN];
static char                     i_curr_zone[ZONENAME_MAX];

/*ARGSUSED*/
static void
brand_warning_func(void *ctx, const char *msg, ...)
 24 brand_error_func(void *ctx, const char *msg, ...)
{
        /*
         * Ignore warning messages from libxml
 27      * Ignore error messages from libxml
         */
}

/*ARGSUSED*/
static void
brand_error_func(void *ctx, const char *msg, ...)
{
        va_list args;

        va_start(args, msg);
        (void) vfprintf(stderr, msg, args);
        va_end(args);
```

```
123 }

125 #endif /* ! codereview */
126 static boolean_t
127 libbrand_initialize()
128 {
129         static mutex_t initialize_lock = DEFAULTMUTEX;

131         (void) mutex_lock(&initialize_lock);

133         if (libbrand_initialized) {
134                 (void) mutex_unlock(&initialize_lock);
135                 return (B_TRUE);
136         }

138         if (sysinfo(SI_ARCHITECTURE, i_curr_arch, sizeof (i_curr_arch)) < 0) {
139                 (void) mutex_unlock(&initialize_lock);
140                 return (B_FALSE);
141         }

143         if (getzonenamebyid(getzoneid(), i_curr_zone,
144             sizeof (i_curr_zone)) < 0) {
145                 (void) mutex_unlock(&initialize_lock);
146                 return (B_FALSE);
147         }

149         /*
150          * Note that here we're initializing per-process libxml2
151          * state.  By doing so we're implicitly assuming that
152          * no other code in this process is also trying to
153          * use libxml2.  But in most case we know this not to
154          * be true since we're almost always used in conjunction
155          * with libzonecfg, which also uses libxml2.  Lucky for
156          * us, libzonecfg initializes libxml2 to essentially
157          * the same defaults as we're using below.
158          */
159         (void) xmlLineNumbersDefault(1);
160         xmlLoadExtDtdDefaultValue |= XML_DETECT_IDS;
161         xmlDoValidityCheckingDefaultValue = 1;
162         (void) xmlKeepBlanksDefault(0);
163         xmlGetWarningsDefaultValue = 0;
164         xmlSetGenericErrorFunc(NULL, brand_error_func);

166         libbrand_initialized = B_TRUE;
167         (void) mutex_unlock(&initialize_lock);
168         return (B_TRUE);
169 }

171 static const char *
172 get_curr_arch(void)
173 {
174         if (!libbrand_initialize())
175                 return (NULL);

177         return (i_curr_arch);
178 }

180 static const char *
181 get_curr_zone(void)
182 {
183         if (!libbrand_initialize())
184                 return (NULL);

186         return (i_curr_zone);
187 }
```



```
189 /*
190  * Internal function to open an XML file
191  *
192  * Returns the XML doc pointer, or NULL on failure.  It will validate the
193  * document, as well as removing any comments from the document structure.
194  */
195 static xmlDocPtr
196 open_xml_file(const char *file)
197 {
198         xmlDocPtr doc;
199         xmlValidCtxtPtr cvp;
200         int valid;

202         if (!libbrand_initialize())
203                 return (NULL);

205         /*
206          * Parse the file
207          */
208         if ((doc = xmlParseFile(file)) == NULL)
209                 return (NULL);

211         /*
212          * Validate the file
213          */
214         if ((cvp = xmlNewValidCtxt()) == NULL) {
215                 xmlFreeDoc(doc);
216                 return (NULL);
217         }
218         cvp->error = brand_error_func;
219         cvp->warning = brand_warning_func;
 31         cvp->warning = brand_error_func;
220         valid = xmlValidateDocument(cvp, doc);
221         xmlFreeValidCtxt(cvp);
222         if (valid == 0) {
223                 xmlFreeDoc(doc);
224                 return (NULL);
225         }

227         return (doc);
228 }
```

_____*unchanged_portion_omitted_*