```
**********************************************************
    9286 Thu Jan 17 14:48:54 2019
new/usr/src/lib/udapl/libdat/common/dat_dictionary.c
10111 dat_dictionary_create() use after free
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License, Version 1.0 only
 * (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) 2002-2003, Network Appliance, Inc. All rights reserved.
 */

/*
 * Copyright 2004 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Copyright (c) 2018, Joyent, Inc.
 */
#pragma ident   "%Z%%M% %I%     %E% SMI"

/*
 *
 * MODULE: dat_dictionary.c
 *
 * PURPOSE: dictionary data structure
 *
 * $Id: dat_dictionary.c,v 1.11 2003/08/05 19:01:48 jlentini Exp $
 */


#include "dat_dictionary.h"


/*
 *
 * Structures
 *
 */

typedef struct DAT_DICTIONARY_NODE
{
    DAT_PROVIDER_INFO           key;
    DAT_DICTIONARY_DATA         data;
    struct DAT_DICTIONARY_NODE  *prev;
    struct DAT_DICTIONARY_NODE  *next;
} DAT_DICTIONARY_NODE;
```

*unchanged_portion_omitted_*


```
/*
 *
 * Function Declarations
 *
 */

static DAT_RETURN
dat_dictionary_key_dup(
    const DAT_PROVIDER_INFO     *old_key,
    DAT_PROVIDER_INFO           *new_key);

static DAT_BOOLEAN
dat_dictionary_key_is_equal(
    const DAT_PROVIDER_INFO     *key_a,
    const DAT_PROVIDER_INFO     *key_b);


/*
 *
 * External Functions
 *
 */


/*
 * Function: dat_dictionary_create
 */

DAT_RETURN
dat_dictionary_create(
    OUT DAT_DICTIONARY **pp_dictionary)
{
        DAT_DICTIONARY  *p_dictionary;
        DAT_RETURN status;

        dat_os_assert(NULL != pp_dictionary);

        status = DAT_SUCCESS;

        /* create the dictionary */
        p_dictionary = dat_os_alloc(sizeof (DAT_DICTIONARY));
        if (NULL == p_dictionary) {
                status = DAT_ERROR(DAT_INSUFFICIENT_RESOURCES,
                    DAT_RESOURCE_MEMORY);
                goto bail;
        }

        (void) dat_os_memset(p_dictionary, '\0', sizeof (DAT_DICTIONARY));

        /* create the head node */
        p_dictionary->head = dat_os_alloc(sizeof (DAT_DICTIONARY_NODE));
        if (NULL == p_dictionary->head) {
                status = DAT_ERROR(DAT_INSUFFICIENT_RESOURCES,
                    DAT_RESOURCE_MEMORY);
                goto bail;
        }

        (void) dat_os_memset(p_dictionary->head, '\0',
            sizeof (DAT_DICTIONARY_NODE));

        /* create the tail node */
        p_dictionary->tail = dat_os_alloc(sizeof (DAT_DICTIONARY_NODE));
        if (NULL == p_dictionary->tail) {
                status = DAT_ERROR(DAT_INSUFFICIENT_RESOURCES,
                    DAT_RESOURCE_MEMORY);
```

```
135                 goto bail;
136         }

138         (void) dat_os_memset(p_dictionary->tail, '\0',
139             sizeof (DAT_DICTIONARY_NODE));

141         p_dictionary->head->next = p_dictionary->tail;
142         p_dictionary->tail->prev = p_dictionary->head;

144         *pp_dictionary = p_dictionary;

146 bail:
147         if (DAT_SUCCESS != status) {
148                 if (NULL != p_dictionary) {
147                         dat_os_free(p_dictionary, sizeof (DAT_DICTIONARY));

149                         if (NULL != p_dictionary->head) {
150                                 dat_os_free(p_dictionary->head,
151                                     sizeof (DAT_DICTIONARY_NODE));
152                         }

154                         if (NULL != p_dictionary->tail) {
155                                 dat_os_free(p_dictionary->tail,
156                                     sizeof (DAT_DICTIONARY_NODE));
157                         }

159                         dat_os_free(p_dictionary, sizeof (DAT_DICTIONARY));
160                 }

162         }

164         return (status);
165 }
```

*unchanged_portion_omitted_*