****
109308 Fri Jun 15 10:07:08 2012
new/usr/src/lib/libzfs/common/libzfs_dataset.c
2883 changing "canmount" property to "on" should not always remount dataset
****

_____unchanged_portion_omitted_

```
1397 /*
1398  * Given a property name and value, set the property for the given dataset.
1399  */
1400 int
1401 zfs_prop_set(zfs_handle_t *zhp, const char *propname, const char *propval)
1402 {
1403         zfs_cmd_t zc = { 0 };
1404         int ret = -1;
1405         prop_changelist_t *cl = NULL;
1406         char errbuf[1024];
1407         libzfs_handle_t *hdl = zhp->zfs_hdl;
1408         nvlist_t *nvl = NULL, *realprops;
1409         zfs_prop_t prop;
1410         boolean_t do_prefix;
1411         uint64_t idx;
1412         int added_resv;

1414         (void) snprintf(errbuf, sizeof (errbuf),
1415             dgettext(TEXT_DOMAIN, "cannot set property for '%s'"),
1416             zhp->zfs_name);

1418         if (nvlist_alloc(&nvl, NV_UNIQUE_NAME, 0) != 0 ||
1419             nvlist_add_string(nvl, propname, propval) != 0) {
1420                 (void) no_memory(hdl);
1421                 goto error;
1422         }

1424         if ((realprops = zfs_valid_proplist(hdl, zhp->zfs_type, nvl,
1425             zfs_prop_get_int(zhp, ZFS_PROP_ZONED), zhp, errbuf)) == NULL)
1426                 goto error;

1428         nvlist_free(nvl);
1429         nvl = realprops;

1431         prop = zfs_name_to_prop(propname);

1433         if (prop == ZFS_PROP_VOLSIZE) {
1434                 if ((added_resv = zfs_add_synthetic_resv(zhp, nvl)) == -1)
1435                         goto error;
1436         }

1438         if ((cl = changelist_gather(zhp, prop, 0, 0)) == NULL)
1439                 goto error;

1441         if (prop == ZFS_PROP_MOUNTPOINT && changelist_haszonedchild(cl)) {
1442                 zfs_error_aux(hdl, dgettext(TEXT_DOMAIN,
1443                     "child dataset with inherited mountpoint is used "
1444                     "in a non-global zone"));
1445                 ret = zfs_error(hdl, EZFS_ZONED, errbuf);
1446                 goto error;
1447         }

1449         /*
1450          * If the dataset's canmount property is being set to noauto,
1451          * or being set to on and the dataset is already mounted,
1452 #endif /* ! codereview */
1453          * then we want to prevent unmounting & remounting it.
1454          */
1455         do_prefix = !((prop == ZFS_PROP_CANMOUNT) &&
```



```
1456             (zprop_string_to_index(prop, propval, &idx,
1457             ZFS_TYPE_DATASET) == 0) && (idx == ZFS_CANMOUNT_NOAUTO ||
1458             (idx == ZFS_CANMOUNT_ON && zfs_is_mounted(zhp, NULL))));
1451             ZFS_TYPE_DATASET) == 0) && (idx == ZFS_CANMOUNT_NOAUTO));

1460         if (do_prefix && (ret = changelist_prefix(cl)) != 0)
1461                 goto error;

1463         /*
1464          * Execute the corresponding ioctl() to set this property.
1465          */
1466         (void) strlcpy(zc.zc_name, zhp->zfs_name, sizeof (zc.zc_name));

1468         if (zcmd_write_src_nvlist(hdl, &zc, nvl) != 0)
1469                 goto error;

1471         ret = zfs_ioctl(hdl, ZFS_IOC_SET_PROP, &zc);

1473         if (ret != 0) {
1474                 zfs_setprop_error(hdl, prop, errno, errbuf);
1475                 if (added_resv && errno == ENOSPC) {
1476                         /* clean up the volsize property we tried to set */
1477                         uint64_t old_volsize = zfs_prop_get_int(zhp,
1478                             ZFS_PROP_VOLSIZE);
1479                         nvlist_free(nvl);
1480                         zcmd_free_nvlists(&zc);
1481                         if (nvlist_alloc(&nvl, NV_UNIQUE_NAME, 0) != 0)
1482                                 goto error;
1483                         if (nvlist_add_uint64(nvl,
1484                             zfs_prop_to_name(ZFS_PROP_VOLSIZE),
1485                             old_volsize) != 0)
1486                                 goto error;
1487                         if (zcmd_write_src_nvlist(hdl, &zc, nvl) != 0)
1488                                 goto error;
1489                         (void) zfs_ioctl(hdl, ZFS_IOC_SET_PROP, &zc);
1490                 }
1491         } else {
1492                 if (do_prefix)
1493                         ret = changelist_postfix(cl);

1495                 /*
1496                  * Refresh the statistics so the new property value
1497                  * is reflected.
1498                  */
1499                 if (ret == 0)
1500                         (void) get_stats(zhp);
1501         }

1503 error:
1504         nvlist_free(nvl);
1505         zcmd_free_nvlists(&zc);
1506         if (cl)
1507                 changelist_free(cl);
1508         return (ret);
1509 }
```

_____unchanged_portion_omitted_