NAME

bectl - Utility to manage boot environments on ZFS

SYNOPSIS

bectl [**-h?**]
bectl [**-r** *beroot*] **activate** [**-t** | **-T**] *beName*
bectl [**-r** *beroot*] **check**
bectl [**-r** *beroot*] **create** [**-r**] [**-e** {*nonActiveBe* | *beName***@***snapshot*}] *newBeName*
bectl [**-r** *beroot*] **create** [**-r**] *beName@snapshot*
bectl [**-r** *beroot*] **destroy** [**-Fo**] *beName*[**@***snapshot*]
bectl [**-r** *beroot*] **export** *sourceBe*
bectl [**-r** *beroot*] **import** *targetBe*
bectl [**-r** *beroot*] **jail** [**-bU**] [{**-o** *key*=*value* | **-u** *key*}]... *beName* [*utility* [*argument* ...]]
bectl [**-r** *beroot*] **list** [**-aDHs**] [**-c** *property*] [**-C** *property*] [{**-c** *property* | **-C** *property*}]
bectl [**-r** *beroot*] **mount** *beName* [*mountpoint*]
bectl [**-r** *beroot*] **rename** *origBeName newBeName*
bectl [**-r** *beroot*] {**ujail** | **unjail**} {*jailId* | *jailName* | *beName*}
bectl [**-r** *beroot*] {**umount** | **unmount**} [**-f**] *beName*

DESCRIPTION

The **bectl** command is used to setup and interact with ZFS boot environments, which are bootable clones of datasets.

A boot environment allows the system to be upgraded, while preserving the pre-upgrade system environment.

bectl itself accepts an **-r** flag specified before the command to indicate the *beroot* that should be used as the boot environment root, or the dataset whose children are all boot environments. Normally this information is derived from the bootfs property of the pool that is mounted at */*, but it is useful when the system has not been booted into a ZFS root or a different pool should be operated on. For instance, booting into the recovery media and manually importing a pool from one of the system's resident disks will require the **-r** flag to work.

Supported Subcommands and Flags

-h | **-?** Print usage information.

activate [**-t** | **-T**] *beName*

Activate the given *beName* as the default boot filesystem. If the **-t** flag is given, this takes effect only for the next boot. Flag **-T** removes temporary boot once configuration. Without temporary configuration, the next boot will use zfs dataset specified in boot pool *bootfs*

property.

check Performs a silent sanity check on the current system. If boot environments are supported and used, **bectl** will exit with a status code of 0. Any other status code is not currently defined and may, in the future, grow special meaning for different degrees of sanity check failures.

create [**-r**] [**-e** {*nonActiveBe* | *beName**@**snapshot*}] *newBeName*
Create a new boot environment named *newBeName*.

If the **-r** flag is given, a recursive boot environment will be made. See *Boot Environment Structures* for a discussion on different layouts.

If the **-e** flag is specified, the new environment will be cloned from the given *nonActiveBe* or *beName**@**snapshot*. Otherwise, the new environment will be created from the currently booted environment.

If **bectl** is creating from another boot environment, a snapshot of that boot environment will be created to clone from.

create [**-r**] *beName@snapshot*
Create a snapshot of the boot environment named *beName*.

If the **-r** flag is given, a recursive snapshot of the boot environment will be created. A snapshot is created for each descendant dataset of the boot environment. See *Boot Environment Structures* for a discussion on different layouts.

No new boot environment is created with this subcommand.

destroy [**-Fo**] *beName*[**@***snapshot*]
Destroy the given *beName* boot environment or *beName**@**snapshot* snapshot without confirmation, unlike in beadm(1). Specifying **-F** will automatically unmount without confirmation.

By default, **bectl** will warn that it is not destroying the origin of *beName*. The **-o** flag may be specified to destroy the origin as well.

export *sourceBe*
Export *sourceBe* to stdout(4). stdout(4) must be piped or redirected to a file.

import *targetBe*

Import *targetBe* from stdin(4).

jail [**-bU**] [{**-o** *key*=*value* | **-u** *key*}]... *beName* [*utility* [*argument* ...]]

Create a jail of the given boot environment. Multiple **-o** and **-u** arguments may be specified. **-o** will set a jail parameter, and **-u** will unset a jail parameter.

By default, jails are created in interactive mode and */bin/sh* is executed within the jail. If *utility* is specified, it will be executed instead of */bin/sh*. The jail will be destroyed and the boot environment unmounted when the command finishes executing, unless the **-U** argument is specified.

The **-b** argument enables batch mode, thereby disabling interactive mode. The **-U** argument will be ignored in batch mode.

The *name*, *host.hostname*, and *path* must be set, the default values are specified below.

All *key*=*value* pairs are interpreted as jail parameters as described in jail(8). The following default parameters are provided:

allow.mount	**true**
allow.mount.devfs	**true**
enforce_statfs	**1**
name	Set to jail ID.
host.hostname	*bootenv*
path	Set to a path in */tmp* generated by libbe(3).

All default parameters may be overwritten.

list [**-aDHs**] [{**-c** *property* | **-C** *property*}]

Display all boot environments. The *Active* field indicates whether the boot environment is active now (*N*); active on reboot (*R*); is used on next boot once (*T*); or combination of (*NRT*).

-a Display all datasets.

-D Display the full space usage for each boot environment, assuming all other boot environments were destroyed.

-H Used for scripting. Do not print headers and separate fields by a single tab instead of arbitrary white space.

-s Display all snapshots as well.

-c *property*
Sort boot environments by the given ZFS dataset property. The following properties are supported:

name (the default)
creation
origin
used
usedbydataset
usedbyrefreservation
usedbysnapshots

Short forms usedds, usedrefreserv and usedsnap are also supported.

-C *property*
Same as the **-c** option, but displays in descending order.

The **-D** option is ignored when either the **-s** or **-a** option is used.

mount *beName* [*mountpoint*]
Mount the given boot environment.

If a nonexistent *mountpoint* is given: **bectl** will make the directory, including intermediate directories as required.

If no *mountpoint* is given: **bectl** will make a directory such as *be_mount.c6Sf* in */tmp*. Randomness in the last four characters of the directory name will prevent mount point conflicts. Unmount of an environment, followed by mount of the same environment without giving a *mountpoint*, will result in a different randomly-named mountpoint.

rename *origBeName newBeName*
Rename the given *origBeName* to the given *newBeName*. The boot environment will not be unmounted in order for this rename to occur.

ujail {*jailId* | *jailName* | *beName*}

unjail {*jailId* | *jailName* | *beName*}
Destroy the jail created from the given boot environment.

umount [**-f**] *beName*

unmount [**-f**] *beName*

Unmount the given boot environment, if it is mounted. Specifying **-f** will force the unmount if busy.

Unmount will not remove the mount point.

Boot Environment Structures

The traditional FreeBSD boot environment layout, as created by the Auto ZFS option to bsdinstall(8), is a "shallow" boot environment structure, where boot environment datasets do not have any directly subordinate datasets. Instead, they’re organized off in *zroot/ROOT*, and they rely on datasets elsewhere in the pool having canmount set to off. For instance, a simplified pool may be laid out as such:

```
% zfs list -o name,canmount,mountpoint
NAME                 CANMOUNT  MOUNTPOINT
zroot
zroot/ROOT           noauto    none
zroot/ROOT/default   noauto    none
zroot/home           on        /home
zroot/usr            off       /usr
zroot/usr/src        on        /usr/src
zroot/var            off       /var
```

In that example, *zroot/usr* has canmount set to off, thus files in */usr* typically fall into the boot environment because this dataset is not mounted. *zroot/usr/src* is mounted, thus files in */usr/src* are not in the boot environment.

The other style of boot environments in use, frequently called "deep boot environments", organizes some or all of the boot environment as subordinate to the boot environment dataset. For example:

```
% zfs list -o name,canmount,mountpoint
NAME                       CANMOUNT  MOUNTPOINT
zroot
zroot/ROOT                 noauto    none
zroot/ROOT/default         noauto    none
zroot/ROOT/default/usr     noauto    /usr
zroot/ROOT/default/usr/local noauto  /usr/local
zroot/var                  on        /var
```

Note that the subordinate datasets now have canmount set to noauto. These are more obviously a part of the boot environment, as indicated by their positioning in the layout. These subordinate datasets will be mounted by the zfsbe rc(8) script at boot time. In this example, */var* is excluded from the boot environment.

bectl subcommands that have their own **-r** operate on this second, "deep" style of boot environment, when the **-r** flag is set. A future version of **bectl** may default to handling both styles and deprecate the various **-r** flags.

SEE ALSO

libbe(3), zfsprops(7), beinstall.sh(8), jail(8), zfs(8), zpool(8)

HISTORY

bectl is based on beadm(1) and was implemented as a project for the 2017 Summer of Code, along with libbe(3).

AUTHORS

bectl was written by Kyle Kneitinger (kneitinger) <*kyle@kneit.in*>.

beadm(1) was written and is maintained by
Slawomir Wojciech Wojtczak (vermaden) <*vermaden@interia.pl*>.

Bryan Drewery (bdrewery) <*bryan@shatow.net*> wrote the original beadm(1) manual page that this one is derived from.