NAME

gpart - control utility for the disk partitioning GEOM class

SYNOPSIS

gpart add -t *type* [**-a** *alignment*] [**-b** *start*] [**-s** *size*] [**-i** *index*] [**-l** *label*] [**-f** *flags*] *geom*
gpart backup *geom*
gpart bootcode [**-N**] [**-b** *bootcode*] [**-p** *partcode* **-i** *index*] [**-f** *flags*] *geom*
gpart commit *geom*
gpart create -s *scheme* [**-n** *entries*] [**-f** *flags*] *provider*
gpart delete -i *index* [**-f** *flags*] *geom*
gpart destroy [**-F**] [**-f** *flags*] *geom*
gpart modify -i *index* [**-l** *label*] [**-t** *type*] [**-f** *flags*] *geom*
gpart recover [**-f** *flags*] *geom*
gpart resize -i *index* [**-a** *alignment*] [**-s** *size*] [**-f** *flags*] *geom*
gpart restore [**-lF**] [**-f** *flags*] *provider* [...]
gpart set -a *attrib* **-i** *index* [**-f** *flags*] *geom*
gpart show [**-l** | **-r**] [**-p**] [*geom* ...]
gpart undo *geom*
gpart unset -a *attrib* **-i** *index* [**-f** *flags*] *geom*
gpart list
gpart status
gpart load
gpart unload

DESCRIPTION

The **gpart** utility is used to partition GEOM providers, normally disks. The first argument is the action to be taken:

add Add a new partition to the partitioning scheme given by *geom*. The partition type must be specified with **-t** *type*. The partition’s location, size, and other attributes will be calculated automatically if the corresponding options are not specified.

The **add** command accepts these options:

-a *alignment* If specified, then the **gpart** utility tries to align *start* offset and partition *size* to be multiple of *alignment* value.

-b *start* The logical block address where the partition will begin. An SI unit suffix is allowed.

-f *flags* Additional operational flags. See the section entitled *OPERATIONAL FLAGS* below for a discussion about its use.

-i *index* The index in the partition table at which the new partition is to be placed. The index determines the name of the device special file used to represent the partition.

-l *label* The label attached to the partition. This option is only valid when used on partitioning schemes that support partition labels.

-s *size* Create a partition of size *size*. An SI unit suffix is allowed.

-t *type* Create a partition of type *type*. Partition types are discussed below in the section entitled *PARTITION TYPES*.

backup Dump a partition table to standard output in a special format used by the **restore** action.

bootcode Embed bootstrap code into the partitioning scheme's metadata on the *geom* (using **-b** *bootcode*) or write bootstrap code into a partition (using **-p** *partcode* and **-i** *index*).

The **bootcode** command accepts these options:

-N Do not preserve the Volume Serial Number for MBR. MBR bootcode contains Volume Serial Number by default, and **gpart** tries to preserve it when installing new bootstrap code. This option skips preservation to help with some versions of boot0cfg(8) that do not support Volume Serial Number.

-b *bootcode* Embed bootstrap code from the file *bootcode* into the partitioning scheme's metadata for *geom*. Not all partitioning schemes have embedded bootstrap code, so the **-b** *bootcode* option is scheme-specific in nature (see the section entitled *BOOTSTRAPPING* below). The *bootcode* file must match the partitioning scheme's requirements for file content and size.

-f *flags* Additional operational flags. See the section entitled *OPERATIONAL FLAGS* below for a discussion about its use.

-i *index* Specify the target partition for **-p** *partcode*.

-p *partcode* Write the bootstrap code from the file *partcode* into the *geom* partition specified by **-i** *index*. The size of the file must be smaller than the size of the partition.

commit Commit any pending changes for geom *geom*. All actions are committed by default and will not result in pending changes. Actions can be modified with the **-f** *flags* option so that they are not committed, but become pending. Pending changes are reflected by the geom and the **gpart** utility, but they are not actually written to disk. The **commit** action will write all pending changes to disk.

create Create a new partitioning scheme on a provider given by *provider*. The scheme to use must be specified with the **-s** *scheme* option.

The **create** command accepts these options:

-f *flags* Additional operational flags. See the section entitled *OPERATIONAL FLAGS* below for a discussion about its use.

-n *entries* The number of entries in the partition table. Every partitioning scheme has a minimum and maximum number of entries. This option allows tables to be created with a number of entries that is within the limits. Some schemes have a maximum equal to the minimum and some schemes have a maximum large enough to be considered unlimited. By default, partition tables are created with the minimum number of entries.

-s *scheme* Specify the partitioning scheme to use. The kernel must have support for a particular scheme before that scheme can be used to partition a disk.

delete Delete a partition from geom *geom* and further identified by the **-i** *index* option. The partition cannot be actively used by the kernel.

The **delete** command accepts these options:

-f *flags* Additional operational flags. See the section entitled *OPERATIONAL FLAGS* below for a discussion about its use.

-i *index* Specifies the index of the partition to be deleted.

destroy Destroy the partitioning scheme as implemented by geom *geom*.

The **destroy** command accepts these options:

-F Forced destroying of the partition table even if it is not empty.

-f *flags* Additional operational flags. See the section entitled *OPERATIONAL FLAGS* below for a discussion about its use.

modify Modify a partition from geom *geom* and further identified by the **-i** *index* option. Only the type and/or label of the partition can be modified. Not all partitioning schemes support labels and it is invalid to try to change a partition label in such cases.

The **modify** command accepts these options:

-f *flags* Additional operational flags. See the section entitled *OPERATIONAL FLAGS* below for a discussion about its use.

-i *index* Specifies the index of the partition to be modified.

-l *label* Change the partition label to *label*.

-t *type* Change the partition type to *type*.

recover Recover a corrupt partition’s scheme metadata on the geom *geom*. See the section entitled *RECOVERING* below for the additional information.

The **recover** command accepts these options:

-f *flags* Additional operational flags. See the section entitled *OPERATIONAL FLAGS* below for a discussion about its use.

resize Resize a partition from geom *geom* and further identified by the **-i** *index* option. If the new size is not specified it is automatically calculated to be the maximum available from *geom*.

The **resize** command accepts these options:

-a *alignment* If specified, then the **gpart** utility tries to align partition *size* to be a multiple of the *alignment* value.

-f *flags* Additional operational flags. See the section entitled *OPERATIONAL FLAGS* below for a discussion about its use.

-i *index* Specifies the index of the partition to be resized.

-s *size* Specifies the new size of the partition, in logical blocks. An SI unit suffix is allowed.

restore Restore the partition table from a backup previously created by the **backup** action and read from standard input. Only the partition table is restored. This action does not affect the content of partitions. After restoring the partition table and writing bootcode if needed, user data must be restored from backup.

The **restore** command accepts these options:

-F Destroy partition table on the given *provider* before doing restore.

-f *flags* Additional operational flags. See the section entitled *OPERATIONAL FLAGS* below for a discussion about its use.

-l Restore partition labels for partitioning schemes that support them.

set Set the named attribute on the partition entry. See the section entitled *ATTRIBUTES* below for a list of available attributes.

The **set** command accepts these options:

-a *attrib* Specifies the attribute to set.

-f *flags* Additional operational flags. See the section entitled *OPERATIONAL FLAGS* below for a discussion about its use.

-i *index* Specifies the index of the partition on which the attribute will be set.

show Show current partition information for the specified geoms, or all geoms if none are specified. The default output includes the logical starting block of each partition, the partition size in blocks, the partition index number, the partition type, and a human readable partition size. Block sizes and locations are based on the device’s Sectorsize as shown by **gpart list**.

The **show** command accepts these options:

-l For partitioning schemes that support partition labels, print them instead of partition type.

-p Show provider names instead of partition indexes.

-r Show raw partition type instead of symbolic name.

undo Revert any pending changes for geom *geom*. This action is the opposite of the **commit** action and can be used to undo any changes that have not been committed.

unset Clear the named attribute on the partition entry. See the section entitled *ATTRIBUTES* below for a list of available attributes.

The **unset** command accepts these options:

-a *attrib* Specifies the attribute to clear.

-f *flags* Additional operational flags. See the section entitled *OPERATIONAL FLAGS* below for a discussion about its use.

-i *index* Specifies the index of the partition on which the attribute will be cleared.

list See geom(8).

status See geom(8).

load See geom(8).

unload See geom(8).

PARTITIONING SCHEMES

Several partitioning schemes are supported by the **gpart** utility:

APM Apple Partition Map, used by PowerPC(R) Macintosh(R) computers. Requires the **GEOM_PART_APM** kernel option.

BSD Traditional BSD disklabel(8), usually used to subdivide MBR partitions. (This scheme can also be used as the sole partitioning method, without an MBR. Partition editing tools from other operating systems often do not understand the bare disklabel partition layout, so this is sometimes called "dangerously dedicated".) Requires the **GEOM_PART_BSD** kernel option.

BSD64 64-bit implementation of BSD disklabel used in DragonFly to subdivide MBR or GPT partitions. Requires the **GEOM_PART_BSD64** kernel option.

LDM The Logical Disk Manager is an implementation of volume manager for Microsoft Windows

NT. Requires the **GEOM_PART_LDM** kernel option.

GPT GUID Partition Table is used on Intel-based Macintosh computers and gradually replacing MBR on most PCs and other systems. Requires the **GEOM_PART_GPT** kernel option.

MBR Master Boot Record is used on PCs and removable media. Requires the **GEOM_PART_MBR** kernel option. The **GEOM_PART_EBR** option adds support for the Extended Boot Record (EBR), which is used to define a logical partition. The **GEOM_PART_EBR_COMPAT** option enables backward compatibility for partition names in the EBR scheme. It also prevents any type of actions on such partitions.

See glabel(8) for additional information on labelization of devices and partitions.

PARTITION TYPES

Partition types are identified on disk by particular strings or magic values. The **gpart** utility uses symbolic names for common partition types so the user does not need to know these values or other details of the partitioning scheme in question. The **gpart** utility also allows the user to specify scheme-specific partition types for partition types that do not have symbolic names. Symbolic names currently understood and used by FreeBSD are:

apple-boot The system partition dedicated to storing boot loaders on some Apple systems. The scheme-specific types are "!171" for MBR, "!Apple_Bootstrap" for APM, and "!426f6f74-0000-11aa-aa11-00306543ecac" for GPT.

bios-boot The system partition dedicated to second stage of the boot loader program. Usually it is used by the GRUB 2 loader for GPT partitioning schemes. The scheme-specific type is "!21686148-6449-6E6F-744E-656564454649".

efi The system partition for computers that use the Extensible Firmware Interface (EFI). The scheme-specific types are "!239" for MBR, and "!c12a7328-f81f-11d2-ba4b-00a0c93ec93b" for GPT.

freebsd A FreeBSD partition subdivided into filesystems with a BSD disklabel. This is a legacy partition type and should not be used for the APM or GPT schemes. The scheme-specific types are "!165" for MBR, "!FreeBSD" for APM, and "!516e7cb4-6ecf-11d6-8ff8-00022d09712b" for GPT.

freebsd-boot A FreeBSD partition dedicated to bootstrap code. The scheme-specific type is "!83bd6b9d-7f41-11dc-be0b-001560b84f0f" for GPT.

freebsd-swap	A FreeBSD partition dedicated to swap space. The scheme-specific types are "!FreeBSD-swap" for APM, and "!516e7cb5-6ecf-11d6-8ff8-00022d09712b" for GPT.
freebsd-ufs	A FreeBSD partition that contains a UFS or UFS2 filesystem. The scheme-specific types are "!FreeBSD-UFS" for APM, and "!516e7cb6-6ecf-11d6-8ff8-00022d09712b" for GPT.
freebsd-vinum	A FreeBSD partition that contains a Vinum volume. The scheme-specific types are "!FreeBSD-Vinum" for APM, and "!516e7cb8-6ecf-11d6-8ff8-00022d09712b" for GPT.
freebsd-zfs	A FreeBSD partition that contains a ZFS volume. The scheme-specific types are "!FreeBSD-ZFS" for APM, and "!516e7cba-6ecf-11d6-8ff8-00022d09712b" for GPT.

Other symbolic names that can be used with the **gpart** utility are:

apple-apfs	An Apple macOS partition used for the Apple file system, APFS.
apple-core-storage	An Apple Mac OS X partition used by logical volume manager known as Core Storage. The scheme-specific type is "!53746f72-6167-11aa-aa11-00306543ecac" for GPT.
apple-hfs	An Apple Mac OS X partition that contains a HFS or HFS+ filesystem. The scheme-specific types are "!175" for MBR, "!Apple_HFS" for APM and "!48465300-0000-11aa-aa11-00306543ecac" for GPT.
apple-label	An Apple Mac OS X partition dedicated to partition metadata that descibes disk device. The scheme-specific type is "!4c616265-6c00-11aa-aa11-00306543ecac" for GPT.
apple-raid	An Apple Mac OS X partition used in a software RAID configuration. The scheme-specific type is "!52414944-0000-11aa-aa11-00306543ecac" for GPT.
apple-raid-offline	An Apple Mac OS X partition used in a software RAID configuration. The scheme-specific type is "!52414944-5f4f-11aa-aa11-00306543ecac" for GPT.
apple-tv-recovery	An Apple Mac OS X partition used by Apple TV. The scheme-specific type is "!5265636f-7665-11aa-aa11-00306543ecac" for GPT.

apple-ufs	An Apple Mac OS X partition that contains a UFS filesystem. The scheme-specific types are "!168" for MBR, "!Apple_UNIX_SVR2" for APM and "!55465300-0000-11aa-aa11-00306543ecac" for GPT.
apple-zfs	An Apple Mac OS X partition that contains a ZFS volume. The scheme-specific type is "!6a898cc3-1dd2-11b2-99a6-080020736631" for GPT. The same GUID is being used also for **illumos/Solaris /usr partition**. See *CAVEATS* section below.
dragonfly-label32	A DragonFly partition subdivided into filesystems with a BSD disklabel. The scheme-specific type is "!9d087404-1ca5-11dc-8817-01301bb8a9f5" for GPT.
dragonfly-label64	A DragonFly partition subdivided into filesystems with a disklabel64. The scheme-specific type is "!3d48ce54-1d16-11dc-8696-01301bb8a9f5" for GPT.
dragonfly-legacy	A legacy partition type used in DragonFly. The scheme-specific type is "!bd215ab2-1d16-11dc-8696-01301bb8a9f5" for GPT.
dragonfly-ccd	A DragonFly partition used with Concatenated Disk driver. The scheme-specific type is "!dbd5211b-1ca5-11dc-8817-01301bb8a9f5" for GPT.
dragonfly-hammer	A DragonFly partition that contains a Hammer filesystem. The scheme-specific type is "!61dc63ac-6e38-11dc-8513-01301bb8a9f5" for GPT.
dragonfly-hammer2	A DragonFly partition that contains a Hammer2 filesystem. The scheme-specific type is "!5cbb9ad1-862d-11dc-a94d-01301bb8a9f5" for GPT.
dragonfly-swap	A DragonFly partition dedicated to swap space. The scheme-specific type is "!9d58fdbd-1ca5-11dc-8817-01301bb8a9f5" for GPT.
dragonfly-ufs	A DragonFly partition that contains an UFS1 filesystem. The scheme-specific type is "!9d94ce7c-1ca5-11dc-8817-01301bb8a9f5" for GPT.
dragonfly-vinum	A DragonFly partition used with Logical Volume Manager. The scheme-specific type is "!9dd4478f-1ca5-11dc-8817-01301bb8a9f5" for GPT.
ebr	A partition subdivided into filesystems with a EBR. The scheme-specific type is "!5" for MBR.
fat16	A partition that contains a FAT16 filesystem. The scheme-specific type is "!6"

for MBR.

fat32	A partition that contains a FAT32 filesystem. The scheme-specific type is "!11" for MBR.
fat32lba	A partition that contains a FAT32 (LBA) filesystem. The scheme-specific type is "!12" for MBR.
hifive-fsbl	A raw partition containing a HiFive first stage bootloader. The scheme-specific type is "!5b193300-fc78-40cd-8002-e86c45580b47" for GPT.
hifive-bbl	A raw partition containing a HiFive second stage bootloader. The scheme-specific type is "!2e54b353-1271-4842-806f-e436d6af6985" for GPT.
linux-data	A Linux partition that contains some filesystem with data. The scheme-specific types are "!131" for MBR and "!0fc63daf-8483-4772-8e79-3d69d8477de4" for GPT.
linux-lvm	A Linux partition dedicated to Logical Volume Manager. The scheme-specific types are "!142" for MBR and "!e6d6d379-f507-44c2-a23c-238f2a3df928" for GPT.
linux-raid	A Linux partition used in a software RAID configuration. The scheme-specific types are "!253" for MBR and "!a19d880f-05fc-4d3b-a006-743f0f84911e" for GPT.
linux-swap	A Linux partition dedicated to swap space. The scheme-specific types are "!130" for MBR and "!0657fd6d-a4ab-43c4-84e5-0933c84b4f4f" for GPT.
mbr	A partition that is sub-partitioned by a Master Boot Record (MBR). This type is known as "!024dee41-33e7-11d3-9d69-0008c781f39f" by GPT.
ms-basic-data	A basic data partition (BDP) for Microsoft operating systems. In the GPT this type is the equivalent to partition types **fat16**, **fat32** and **ntfs** in MBR. This type is used for GPT exFAT partitions. The scheme-specific type is "!ebd0a0a2-b9e5-4433-87c0-68b6b72699c7" for GPT.
ms-ldm-data	A partition that contains Logical Disk Manager (LDM) volumes. The scheme-specific types are "!66" for MBR, "!af9b60a0-1431-4f62-bc68-3311714a69ad" for GPT.

ms-ldm-metadata	A partition that contains Logical Disk Manager (LDM) database. The scheme-specific type is "!5808c8aa-7e8f-42e0-85d2-e1e90434cfb3" for GPT.
netbsd-ccd	A NetBSD partition used with Concatenated Disk driver. The scheme-specific type is "!2db519c4-b10f-11dc-b99b-0019d1879648" for GPT.
netbsd-cgd	An encrypted NetBSD partition. The scheme-specific type is "!2db519ec-b10f-11dc-b99b-0019d1879648" for GPT.
netbsd-ffs	A NetBSD partition that contains an UFS filesystem. The scheme-specific type is "!49f48d5a-b10e-11dc-b99b-0019d1879648" for GPT.
netbsd-lfs	A NetBSD partition that contains an LFS filesystem. The scheme-specific type is "!49f48d82-b10e-11dc-b99b-0019d1879648" for GPT.
netbsd-raid	A NetBSD partition used in a software RAID configuration. The scheme-specific type is "!49f48daa-b10e-11dc-b99b-0019d1879648" for GPT.
netbsd-swap	A NetBSD partition dedicated to swap space. The scheme-specific type is "!49f48d32-b10e-11dc-b99b-0019d1879648" for GPT.
ntfs	A partition that contains a NTFS or exFAT filesystem. The scheme-specific type is "!7" for MBR.
prep-boot	The system partition dedicated to storing boot loaders on some PowerPC systems, notably those made by IBM. The scheme-specific types are "!65" for MBR and "!9e1a2d38-c612-4316-aa26-8b49521e5a8b" for GPT.
solaris-boot	A illumos/Solaris partition dedicated to boot loader. The scheme-specific type is "!6a82cb45-1dd2-11b2-99a6-080020736631" for GPT.
solaris-root	A illumos/Solaris partition dedicated to root filesystem. The scheme-specific type is "!6a85cf4d-1dd2-11b2-99a6-080020736631" for GPT.
solaris-swap	A illumos/Solaris partition dedicated to swap. The scheme-specific type is "!6a87c46f-1dd2-11b2-99a6-080020736631" for GPT.
solaris-backup	A illumos/Solaris partition dedicated to backup. The scheme-specific type is "!6a8b642b-1dd2-11b2-99a6-080020736631" for GPT.

solaris-var A illumos/Solaris partition dedicated to /var filesystem. The scheme-specific type is "!6a8ef2e9-1dd2-11b2-99a6-080020736631" for GPT.

solaris-home A illumos/Solaris partition dedicated to /home filesystem. The scheme-specific type is "!6a90ba39-1dd2-11b2-99a6-080020736631" for GPT.

solaris-altsec A illumos/Solaris partition dedicated to alternate sector. The scheme-specific type is "!6a9283a5-1dd2-11b2-99a6-080020736631" for GPT.

solaris-reserved A illumos/Solaris partition dedicated to reserved space. The scheme-specific type is "!6a945a3b-1dd2-11b2-99a6-080020736631" for GPT.

vmware-vmfs A partition that contains a VMware File System (VMFS). The scheme-specific types are "!251" for MBR and "!aa31e02a-400f-11db-9590-000c2911d1b8" for GPT.

vmware-vmkdiag A partition that contains a VMware diagostic filesystem. The scheme-specific types are "!252" for MBR and "!9d275380-40ad-11db-bf97-000c2911d1b8" for GPT.

vmware-reserved A VMware reserved partition. The scheme-specific type is "!9198effc-31c0-11db-8f-78-000c2911d1b8" for GPT.

vmware-vsanhdr A partition claimed by VMware VSAN. The scheme-specific type is "!381cfccc-7288-11e0-92ee-000c2911d0b2" for GPT.

ATTRIBUTES

The scheme-specific attributes for EBR:

active

The scheme-specific attributes for GPT:

bootme When set, the **gptboot** stage 1 boot loader will try to boot the system from this partition. Multiple partitions can be marked with the **bootme** attribute. See gptboot(8) for more details.

bootonce Setting this attribute automatically sets the **bootme** attribute. When set, the **gptboot** stage 1 boot loader will try to boot the system from this partition only once. Multiple partitions can be marked with the **bootonce** and **bootme** attribute pairs. See gptboot(8) for more details.

bootfailed This attribute should not be manually managed. It is managed by the **gptboot** stage 1 boot loader and the */etc/rc.d/gptboot* start-up script. See gptboot(8) for more details.

lenovofix Setting this attribute overwrites the Protective MBR with a new one where the 0xee partition is the second, rather than the first record. This resolves a BIOS compatibility issue with some Lenovo models including the X220, T420, and T520, allowing them to boot from GPT partitioned disks without using EFI.

The scheme-specific attributes for MBR:

active

BOOTSTRAPPING

FreeBSD supports several partitioning schemes and each scheme uses different bootstrap code. The bootstrap code is located in a specific disk area for each partitioning scheme, and may vary in size for different schemes.

Bootstrap code can be separated into two types. The first type is embedded in the partitioning scheme's metadata, while the second type is located on a specific partition. Embedding bootstrap code should only be done with the **gpart bootcode** command with the **-b** *bootcode* option. The GEOM PART class knows how to safely embed bootstrap code into specific partitioning scheme metadata without causing any damage.

The Master Boot Record (MBR) uses a 512-byte bootstrap code image, embedded into the partition table's metadata area. There are two variants of this bootstrap code: */boot/mbr* and */boot/boot0*. */boot/mbr* searches for a partition with the **active** attribute (see the *ATTRIBUTES* section) in the partition table. Then it runs next bootstrap stage. The */boot/boot0* image contains a boot manager with some additional interactive functions for multi-booting from a user-selected partition.

A BSD disklabel is usually created inside an MBR partition (slice) with type **freebsd** (see the *PARTITION TYPES* section). It uses 8 KB size bootstrap code image */boot/boot*, embedded into the partition table's metadata area.

Both types of bootstrap code are used to boot from the GUID Partition Table. First, a protective MBR is embedded into the first disk sector from the */boot/pmbr* image. It searches through the GPT for a **freebsd-boot** partition (see the *PARTITION TYPES* section) and runs the next bootstrap stage from it. The **freebsd-boot** partition should be smaller than 545 KB. It can be located either before or after other FreeBSD partitions on the disk. There are two variants of bootstrap code to write to this partition: */boot/gptboot* and */boot/gptzfsboot*.

/boot/gptboot is used to boot from UFS partitions. **gptboot** searches through **freebsd-ufs** partitions in the GPT and selects one to boot based on the **bootonce** and **bootme** attributes. If neither attribute is found, */boot/gptboot* boots from the first **freebsd-ufs** partition. */boot/loader* (the third bootstrap stage) is loaded from the first partition that matches these conditions. See gptboot(8) for more information.

/boot/gptzfsboot is used to boot from ZFS. It searches through the GPT for **freebsd-zfs** partitions, trying to detect ZFS pools. After all pools are detected, */boot/loader* is started from the first one found set as bootable.

The APM scheme also does not support embedding bootstrap code. Instead, the 800 KBytes bootstrap code image */boot/boot1.hfs* should be written with the **gpart bootcode** command to a partition of type **apple-boot**, which should also be 800 KB in size.

OPERATIONAL FLAGS

Actions other than the **commit** and **undo** actions take an optional **-f** *flags* option. This option is used to specify action-specific operational flags. By default, the **gpart** utility defines the ‘C’ flag so that the action is immediately committed. The user can specify "**-f x**" to have the action result in a pending change that can later, with other pending changes, be committed as a single compound change with the **commit** action or reverted with the **undo** action.

RECOVERING

The GEOM PART class supports recovering of partition tables only for GPT. The GPT primary metadata is stored at the beginning of the device. For redundancy, a secondary (backup) copy of the metadata is stored at the end of the device. As a result of having two copies, some corruption of metadata is not fatal to the working of GPT. When the kernel detects corrupt metadata, it marks this table as corrupt and reports the problem. **destroy** and **recover** are the only operations allowed on corrupt tables.

If one GPT header appears to be corrupt but the other copy remains intact, the kernel will log the following:

```
GEOM: provider: the primary GPT table is corrupt or invalid.
GEOM: provider: using the secondary instead -- recovery strongly advised.
```

or

```
GEOM: provider: the secondary GPT table is corrupt or invalid.
GEOM: provider: using the primary only -- recovery suggested.
```

Also **gpart** commands such as **show**, **status** and **list** will report about corrupt tables.

If the size of the device has changed (e.g., volume expansion) the secondary GPT header will no longer be located in the last sector. This is not a metadata corruption, but it is dangerous because any corruption of the primary GPT will lead to loss of the partition table. This problem is reported by the kernel with the message:

```
GEOM: provider: the secondary GPT header is not in the last LBA.
```

This situation can be recovered with the **recover** command. This command reconstructs the corrupt metadata using known valid metadata and relocates the secondary GPT to the end of the device.

NOTE: The GEOM PART class can detect the same partition table visible through different GEOM providers, and some of them will be marked as corrupt. Be careful when choosing a provider for recovery. If you choose incorrectly you can destroy the metadata of another GEOM class, e.g., GEOM MIRROR or GEOM LABEL.

SYSCTL VARIABLES

The following sysctl(8) variables can be used to control the behavior of the **PART** GEOM class. The default value is shown next to each variable.

kern.geom.part.allow_nesting: 0
: By default, some schemes (currently BSD and BSD64) do not permit further nested partitioning. This variable overrides this restriction and allows arbitrary nesting (except within partitions created at offset 0). Some schemes have their own separate checks, for which see below.

kern.geom.part.auto_resize: 1
: This variable controls automatic resize behavior of the **PART** GEOM class. When this variable is enable and new size of provider is detected, the schema metadata is resized but all changes are not saved to disk, until **gpart commit** is run to confirm changes. This behavior is also reported with diagnostic message: **GEOM_PART: (provider) was automatically resized. Use ‘gpart commit (provider)‘ to save changes or ‘gpart undo (provider)‘ to revert them.**

kern.geom.part.check_integrity: 1
: This variable controls the behaviour of metadata integrity checks. When integrity checks are enabled, the **PART** GEOM class verifies all generic partition parameters obtained from the disk metadata. If some inconsistency is detected, the partition table will be rejected with a diagnostic message: **GEOM_PART: Integrity check failed (provider, scheme)**.

kern.geom.part.gpt.allow_nesting: 0
: By default the GPT scheme is allowed only at the outermost nesting level. This variable allows this restriction to be removed.

kern.geom.part.ldm.debug: 0
Debug level of the Logical Disk Manager (LDM) module. This can be set to a number between 0 and 2 inclusive. If set to 0 minimal debug information is printed, and if set to 2 the maximum amount of debug information is printed.

kern.geom.part.ldm.show_mirrors: 0
This variable controls how the Logical Disk Manager (LDM) module handles mirrored volumes. By default mirrored volumes are shown as partitions with type **ms-ldm-data** (see the *PARTITION TYPES* section). If this variable set to 1 each component of the mirrored volume will be present as independent partition. *NOTE*: This may break a mirrored volume and lead to data damage.

kern.geom.part.mbr.enforce_chs: 0
Specify how the Master Boot Record (MBR) module does alignment. If this variable is set to a non-zero value, the module will automatically recalculate the user-specified offset and size for alignment with the CHS geometry. Otherwise the values will be left unchanged.

kern.geom.part.separator:
Specify an optional separator that will be inserted between the GEOM name and partition name. This variable is a loader(8) tunable. Note that setting this variable may break software which assumes a particular naming scheme.

EXIT STATUS

Exit status is 0 on success, and 1 if the command fails.

EXAMPLES

The examples below assume that the disk’s logical block size is 512 bytes, regardless of its physical block size.

GPT

In this example, we will format *ada0* with the GPT scheme and create boot, swap and root partitions. First, we need to create the partition table:

```
/sbin/gpart create -s GPT ada0
```

Next, we install a protective MBR with the first-stage bootstrap code. The protective MBR lists a single, bootable partition spanning the entire disk, thus allowing non-GPT-aware BIOSes to boot from the disk and preventing tools which do not understand the GPT scheme from considering the disk to be unformatted.

```
/sbin/gpart bootcode -b /boot/pmbr ada0
```

We then create a dedicated **freebsd-boot** partition to hold the second-stage boot loader, which will load the FreeBSD kernel and modules from a UFS or ZFS filesystem. This partition must be larger than the bootstrap code (either */boot/gptboot* for UFS or */boot/gptzfsboot* for ZFS), but smaller than 545 kB since the first-stage loader will load the entire partition into memory during boot, regardless of how much data it actually contains. We create a 472-block (236 kB) boot partition at offset 40, which is the size of the partition table (34 blocks or 17 kB) rounded up to the nearest 4 kB boundary.

```
/sbin/gpart add -b 40 -s 472 -t freebsd-boot ada0
/sbin/gpart bootcode -p /boot/gptboot -i 1 ada0
```

We now create a 4 GB swap partition at the first available offset, which is 40 + 472 = 512 blocks (256 kB).

```
/sbin/gpart add -s 4G -t freebsd-swap ada0
```

Aligning the swap partition and all subsequent partitions on a 256 kB boundary ensures optimal performance on a wide range of media, from plain old disks with 512-byte blocks, through modern "advanced format" disks with 4096-byte physical blocks, to RAID volumes with stripe sizes of up to 256 kB.

Finally, we create and format an 8 GB **freebsd-ufs** partition for the root filesystem, leaving the rest of the device free for additional filesystems:

```
/sbin/gpart add -s 8G -t freebsd-ufs ada0
/sbin/newfs -Uj /dev/ada0p3
```

MBR

In this example, we will format *ada0* with the MBR scheme and create a single partition which we subdivide using a traditional BSD disklabel.

First, we create the partition table as well as a single partition 64 GB in size and an alignment of 4 kB, then we mark that partition active (bootable) and install the first-stage boot loader:

```
/sbin/gpart create -s MBR ada0
/sbin/gpart add -t freebsd -s 64G -a 4k ada0
/sbin/gpart set -a active -i 1 ada0
/sbin/gpart bootcode -b /boot/boot0 ada0
```

Next, we create a disklabel in that partition ("slice" in disklabel terminology) with room for up to 20 partitions:

```
/sbin/gpart create -s BSD -n 20 ada0s1
```

We then create an 8 GB root partition and a 4 GB swap partition:

```
/sbin/gpart add -t freebsd-ufs -s 8G ada0s1
/sbin/gpart add -t freebsd-swap -s 4G ada0s1
```

Finally, we install the appropriate boot loader for the BSD label:

```
/sbin/gpart bootcode -b /boot/boot ada0s1
```

Deleting Partitions and Destroying the Partitioning Scheme

If a *Device busy* error is shown when trying to destroy a partition table, remember that all of the partitions must be deleted first with the **delete** action. In this example, *da0* has three partitions:

```
/sbin/gpart delete -i 3 da0
/sbin/gpart delete -i 2 da0
/sbin/gpart delete -i 1 da0
/sbin/gpart destroy da0
```

Rather than deleting each partition and then destroying the partitioning scheme, the **-F** option can be given with **destroy** to delete all of the partitions before destroying the partitioning scheme. This is equivalent to the previous example:

```
/sbin/gpart destroy -F da0
```

Backup and Restore

Create a backup of the partition table from *da0*:

```
/sbin/gpart backup da0 > da0.backup
```

Restore the partition table from the backup to *da0*:

```
/sbin/gpart restore -l da0 < /mnt/da0.backup
```

Clone the partition table from *ada0* to *ada1* and *ada2*:

```
      /sbin/gpart backup ada0 | /sbin/gpart restore -F ada1 ada2
```

SEE ALSO

geom(4), boot0cfg(8), geom(8), glabel(8), gptboot(8)

HISTORY

The **gpart** utility appeared in FreeBSD 7.0.

AUTHORS

Marcel Moolenaar <*marcel@FreeBSD.org*>

CAVEATS

Partition type *apple-zfs* (6a898cc3-1dd2-11b2-99a6-080020736631) is also being used on illumos/Solaris platforms for ZFS volumes.