

# Secrets of the GPT

Technical Note TN2166

(Apple Computer)

Apple has switched to a new disk partitioning scheme known as the GUID partition table, or GPT. This new scheme offers a number of advantages over the previous scheme, but it also presents some new challenges. This technote describes GPT in general, and gives some specific details about how Apple uses GPT.

You should read this technote if you're writing software that interacts with Apple's implementation of GPT. Specifically, this technote is vital if you're writing a disk utility for Mac OS X, or trying to run some alternative operating system, like Linux, on a Macintosh computer.

## The Road to GPT

Apple's previous disk partitioning scheme, known as Apple partition map (APM), was introduced with the Macintosh II in 1987. The scheme was very well designed, and it has survived, with very few changes, for almost twenty years.

However, in recent years APM's limitations have been looming on the horizon. Specifically, APM is restricted to 32-bits worth of blocks. With a standard block size of 512 bytes, this translates to a maximum disk size of 2 TB. With the rate that hard disks are growing, it's easy to imagine a typical desktop computer shipping with more than 2 TB of storage in the next few years.

Apple did consider extending APM to support larger disks. However, as such a change would break all existing partitioning tools, it was just as convenient to switch to an entirely new partitioning scheme. After some serious thought, Apple decided to adopt GPT.

The GUID partition table (GPT) partitioning scheme was introduced by Intel as part of an effort to introduce more modern firmware to generic PC hardware. Traditional PC hardware uses BIOS firmware, which uses a partitioning scheme known as master boot record (MBR). MBR has lots of severe limitations, and is not appropriate for a modern computer. Intel's modern firmware, known as the Extensible Firmware Interface (EFI), includes a new partitioning scheme, GPT.

One of the nice features about GPT is that it is defined by a formal standard. Specifically, GPT is defined by the following documents:

- Section 11.2.2 of "Extensible Firmware Interface Specification", version 1.1, available from Intel
- Chapter 5 "GUID Partition Table (GPT) Format" of the "Unified Extensible Firmware Interface Specification", version 2.0, available from the Unified EFI Forum

Because GPT is defined by a formal specification, it is not necessary to describe it in detail here. Thus, this technote concentrates on how GPT is used by Apple. It starts with a high-level overview of GPT. It then describes how Apple has rolled out support for GPT, including a discussion of our partitioning policies. Finally, it concludes with some hints and tips for working with GPT on the Macintosh.

## GPT Overview

GPT is a simple but effective partitioning scheme. Table 1 shows a high-level summary of the block layout used by GPT. In this table,  $n$  is the number of blocks on the disk, and  $b$  is the number of blocks used to describe the partition entry array (typically 32, but see below for a discussion of this).

# Secrets of the GPT

Technical Note TN2166

(Apple Computer)

**Table 1**  
**GPT Summary**

Block	Description
0	Protective MBR
1	Partition Table Header (primary)
2 through 2+b-1	Partition Entry Array (primary)
2+b through n-2-b	partition data
n-2-b+1 through n-2	Partition Entry Array (backup)
n-1	Partition Table Header (backup)

The protective MBR is an MBR that defines a single partition entry that covers the entire area of the disk used by GPT structures and partitions. It is designed to prevent GPT-unaware programs from accidentally modifying a GPT disk. A GPT-unaware program sees the GPT disk as an MBR disk with a single, unknown partition. In a way, this is like the HFS wrapper around an HFS Plus disk.

**WARNING:** See Protective MBR for an important discussion of the protective MBR.

The partition table header is a structure that defines various aspects of the disk, including a GUID to uniquely identify the disk, the starting block of the partition entry array, and the size of each partition entry in that array.

The partition entry array is an array of partition entries, each of which defines a partition (or is all zero, indicating that the entry is not in use). The array is stored in a contiguous range of blocks on disk, which is treated as an array of bytes. The first partition entry starts at the first byte of the array, the next partition entry follows immediately after that, and so on. The size of these entries is given by a field in the partition table header.

Each partition entry contains a GUID to uniquely identify the partition itself, a GUID to identify the partition type, the start and end block of the partition, and the partition name.

For more details about these structures, read the GPT specification. As you're reading, please keep in mind the following:

- GPT is little endian; that is, multi-byte quantities are stored with the least significant byte first.
- In GPT, GUIDs are considered to have a structure, and you must treat the structured parts as little

# Secrets of the GPT

Technical Note TN2166

(Apple Computer)

endian. For example, the GUID for the ESP partition type is shown in the documentation as C12A7328-F81F-11D2-BA4B-00A0C93EC93B, but the bytes written to disk are 0x28 0x73 0x2a 0xc1 0x1f 0xf8 0xd2 0x11 0xba 0x4b 0x00 0xa0 0xc9 0x3e 0xc9 0x3b. That is, the first three dash-delimited fields of the GUID are stored little endian, and the last two fields are not. See Appendix A of the EFI specification for more details.

- When one GPT structure points to another, it does so via a block number (in the terms used by the specification, this is a logical block address, or LBA). To correlate this to a byte offset (for example, if you're seeking within a device node), you must multiply it by the block size. You can get the block size of a device node by calling `ioctl` with the `DKIOCGETBLOCKSIZE` selector (defined in `<sys/disk.h>`). Do not assume that the block size is always going to be 512 bytes.
- GPT requires that the partition entry array be at least 16 KB long. For a 512 byte block size, this translates to 32 blocks. For a 4096 byte block size, this is only 4 blocks. This is a minimum value; the partition entry array can be longer.
- When iterating over the partition entry array, it is vital that you use the `SizeOfPartitionEntry` field of the partition table header to determine the size of the entries. Do not hardwire the current size of the partition entry (128 bytes).
- Unlike APM, there is no requirement that every block be covered by a partition. Any block between the first and last usable block (denoted by the `FirstUsableLBA` and `LastUsableLBA` fields of the partition table header) that is not covered by a valid partition is considered to be free. Thus, there is no equivalent to the `Apple_Free` partition type.
- There is no requirement that partition entries be sorted in the partition entry array.

## Apple's Support for GPT

Any Macintosh computer running Mac OS X 10.4 and later can mount GPT-partitioned disks.

Intel-based Macintosh computers can boot from GPT. By default, the internal hard disk is formatted as GPT.

**IMPORTANT:** While Intel-based Macintosh computers can boot from GPT and APM, Apple only supports booting Mac OS X on these machines from GPT. Apple's GUI tools, like the Installer, will prevent you installing Mac OS X for an Intel-based Mac on non-GPT disks.

On all Intel-based Macintosh computers (and, starting with Mac OS X 10.4.6, PowerPC-based computers as well), Disk Utility has full support for GPT. The `diskutil` command line tool also includes GPT support. See its man page for details.

Note: If you're using a PowerPC-based computer running Mac OS X 10.4.6, you'll be disappointed to discover that the `diskutil` man page has not been updated for GPT. Rest assured that the actual `diskutil` code is GPT-aware. Use the tool's built-in help to learn how to create a GPT disk.

## Advice for Partitioners

The GPT specification is very flexible; the standard does not define exactly how to partition a disk, it merely describes the realm of what's possible. When Apple implemented our support for GPT, we made a number of policy decisions on how to lay out the disk, and we recommend that third party disk utilities for the Macintosh follow our policies.

# Secrets of the GPT

Technical Note TN2166

(Apple Computer)

## GPT Partition Types

Most GPT partition types are defined in the GPT specification. Beyond that, Apple has defined a number of GUIDs to denote Apple-specific partition types. These are described in Table 2.

**Table 2**  
**Apple-defined GPT Partition Types**

Type	GUID	APM Equivalent
HFS [Plus]	48465300-0000-11AA-AA11-00306543ECAC	Apple_HFS
UFS	55465300-0000-11AA-AA11-00306543ECAC	Apple_UFS
Boot	426F6F74-0000-11AA-AA11-00306543ECAC	Apple_Boot
RAID	52414944-0000-11AA-AA11-00306543ECAC	Apple_RAID
Offline RAID	52414944-5F4F-11AA-AA11-00306543ECAC	Apple_RAID_Offline
Label	4C616265-6C00-11AA-AA11-00306543ECAC	Apple_Label

## Partitioning Policy

When partitioning a disk as GPT, we lay out the partition map differently depending on whether the disk is tiny, small or big. For the sake of this discussion, a tiny disk is one whose size is less than 1 GB, a small disk is between 1 GB and 2 GB in size, and a big disk is one whose size is 2 GB or more.

Apple's partitioning policy is as follows:

- For all types of disks, we align each partition to a 4 KB boundary.
- For tiny disks, we create the partitions more-or-less as the user specified. That is, we create no extra partitions and we reserve no significant amount of free space.
- For small disks, we create no extra partitions, but we do leave 128MB of free space after each partition.
- For big disks, we always create a 200 MB EFI system partition (ESP) (see ESP Explained) as the first partition on the disk. We also leave 128 MB of free space after each partition (not including the ESP).

Table 3 is a summary of the policy for each disk type.

# Secrets of the GPT

Technical Note TN2166

(Apple Computer)

**Table 3: Partitioning Policy by Disk Type**

Type	Size Range	Partition Alignment	Free Space	ESP
Tiny	0 <= size < 1 GB	4 KB	no	none
Small	1 GB <= size < 2 GB	4 KB	128 MB after each partition	none
Big	2GB <= size	4 KB	128 MB after each partition (except the ESP)	200 MB

**IMPORTANT:** We align partitions to 4 KB to accommodate limitations of the HFS Plus file system implementation on Mac OS X (r. 4259751).

Note: We leave free space after each partition to make it easier for future system software to manipulate the partition map in ways that we can't anticipate currently.

## ESP Explained

The EFI system partition (ESP) is a special partition from which EFI can load EFI (boot-time) device drivers. The EFI firmware in Macintosh computers fully supports the ESP, although Apple does not currently use it for anything. We create the ESP on big disks to make things easier if we ever need to use ESP-based drivers. We strongly recommend that you do the same.

When you initialise the ESP, keep in mind that it must conform to the specification in the EFI documentation. While the ESP looks like a FAT32 volume, it is actually an EFI file system which "will be maintained by specification and will not evolve over time to deal with errata or variant interpretations in OS file system drivers". For more details, see Section 12.2 "File System Format" of the UEFI Specification, version 2.0.

Thus, you should not use the `newfs_msdos` utility to create the ESP because there is no way to guarantee that it will create an EFI file system. You might consider including a (compressed) disk image of the ESP within your disk utility.

## Hints and Tips

This section describes a number of helpful hints when dealing with GPT.

## GPT Command Line Tools

Since Mac OS X 10.4, Apple has included a command line tool, `gpt`, for exploring and modifying GPT disks. You can learn more about this tool by reading its man page. The source for this tool is included in Darwin.

Systems that fully support GPT include a version of the `diskutil` that understands GPT.

# Secrets of the GPT

Technical Note TN2166

(Apple Computer)

Be aware that these two tools are based on different source bases. The `gpt` tool is a standard-alone program derived from the FreeBSD `gpt` tool, and its source is in Darwin. The `diskutil` tool is an Apple-created program that uses Apple-private infrastructure (the Disk Management framework and, underlying that, the MediaKit framework) to do its job.

**IMPORTANT:** Because of its heritage, the `gpt` tool does not implement the partitioning policy described in Partitioning Policy.

One nice feature of the `diskutil` tool is that it will display known GPT partition types in a human-readable form, rather than as a raw GUID.

## Copying Concerns

Be careful when doing a block-for-block copy of a GPT disk. The GUID in the partition table header that identifies the disk (and the GUIDs in each partition entry) are meant to be globally unique, and Apple's system software relies on this feature. If you block copy a disk, you should consider whether it's appropriate to set these GUIDs to new values. For example:

- If you're block copying a disk for the purposes of backup (that is, the copy is stored offline and can only ever be restored on top of the original), you should not update the GUIDs in the GPT. That way, when you restore the disk, it is completely identical to the original.
- On the other hand, if you're duplicating the disk (that is, you're copying the contents of one disk to another), you should update the GUIDs in the GPT. That way the system will see the original and the copy as different disks.

If you don't follow this advice, you may encounter weird problems. For example, if you duplicate a disk without updating the GUIDs, the computer might boot from either the original or the copy in an unpredictable fashion (perhaps toggling from boot to boot).

## The Last Block

There are a number of third party disk drives that, due to buggy firmware, report an error if you access the last block on the disk. This was not a serious problem for APM. It's easy to create an APM that avoids the last block by creating a small `Apple_Free` partition at the end of the disk (this is what Apple's implementation of APM does).

For GPT, however, this problem is much more serious. The GPT specification requires that the last block of the disk contain the backup partition table header; you can't just choose to avoid it!

Apple handles this problem by ignoring any error when reading the backup partition table header in the last block of a GPT disk. If such an error occurs, the disk has a valid primary partition table and an invalid backup. According to the rules laid out by the GPT specification, such a disk is still considered to be a valid GPT disk.

On the other hand, when partitioning a disk as GPT, if writing to the last block results in an error, the partitioning operation will fail. Such a disk can't be partitioned as a GPT disk using Apple software.

## Protective MBR

By definition, all GPT disks start with a protective MBR. This is an MBR that defines a single partition entry (of type `0xEE`) that covers the entire area of the disk used by GPT structures and partitions. It is designed to prevent GPT-unaware programs from accidentally modifying a GPT disk.

# Secrets of the GPT

Technical Note TN2166

(Apple Computer)

A GPT-aware program can recognise that a disk contains a GPT by checking block 1 and the last block, as discussed in the EFI specification. In addition, it should check that block 0 contains a protective MBR. If block 0 contains any other form of MBR, it should refuse to manipulate the disk. Specifically, if block 0 contains an MBR with more than one partition entry, or a single partition entry whose OSType is not 0xEE, it is not a compliant GPT disk, and manipulating the GPT may cause dangerous inconsistencies between it and the legacy MBR.

**WARNING:** Failure to comply with this recommendation may result in the loss of user data.