```
vdevno = 2, size = 5120, dsync = true
```

Usage

- The master device is initialized by the installation program; you need not initialize this device with **disk init**.
- To successfully complete disk initialization, the "sybase" user must have the appropriate operating system permissions on the device that is being initialized.
- You can specify the size as a float, but the size is rounded down to the nearest multiple of 2K.
- If you do not use a unit specifier for *size*:
  - **disk init** uses the virtual page size of 2K.
  - The *size* argument for create database and alter database is in terms of megabytes of disk space. This value is converted to the number of logical pages the master device was built with
- The minimum size of a disk piece that you can initialize using **disk init** is the larger of:
  - One megabyte
  - One allocation unit of the server's logical page size

8.10.2

- Use **disk init** for each new database device. Each time **disk init** is issued, a row is added to *master..sysdevices*. A new database device does not automatically become part of the pool of default database storage. Assign default status to a database device with **sp_diskdefault**.
- Back up the *master* database with the dump database or dump transaction command after each use of **disk init**. This makes recovery easier and safer in case *master* is damaged. If you add a device with **disk init** and fail to back up *master*, you may be able to recover the changes by using disk reinit, then stopping and restarting Adaptive Server.
- Assign user databases to database devices with the **name** clause of the create database or alter database command.
- The preferred method for placing a database's transaction log (the system table *syslogs*) on a different device than the one on which the rest of the database is stored, is to use the **log on** extension to create database. Alternatively, you can name at least two devices when you create the database, then execute **sp_logdevice**. You can also use alter database to extend the database onto a second device, then run **sp_logdevice**. The **log on** extension immediately moves the entire log to a separate device. The **sp_logdevice** method retains part of the system log on the original database device until transaction activity causes the migration to become complete.
- For a report on all Adaptive Server devices on your system (both database and dump devices), execute **sp_helpdevice**.
- Remove a database device with **sp_dropdevice**. You must first drop all existing databases on that device.
- If **disk unit** failed because the **size** value is too large for the database device, use a different virtual device number or restart Adaptive Server before executing **disk unit** again.

Using dsync

Do not set **dsync** to **false** for any device that stores critical data. The only exception is *tempdb*, which can safely be stored on devices for which **dsync** is set to **false**.

- When **dsync** is on, writes to the database device are guaranteed to take place on the physical storage media, and Adaptive Server can recover data on the device in the event of a system failure.
- When **dsync** is off, writes to the database device may be buffered by the UNIX file system. The UNIX file system may mark an update as being completed, even though the physical media has not yet been modified. In the event of a system failure, there is no guarantee that data updates have ever taken place on the physical media, and Adaptive Server may be unable to recover the database.
- **dsync** is always on for the master device file.
- The **dsync** value should be turned off only when databases on the device need not be recovered after a system failure. For example, you may consider turning **dsync** off for a device that stores only the *tempdb* database.
- Adaptive Server ignores the **dsync** setting for devices stored on raw partitions—writes to those device are guaranteed to take place on the physical storage media, regardless of the **dsync** setting.
- The **dsync** setting is not used on the Windows NT platform.
- **disk reinit** ensures that *master..sysdevices* is correct if the **master** database has been damaged or if devices have been added since the last dump of **master**.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

8.10.2

Permissions