SetResPurge Force resource changes to be written before purge

#include <Resources.h> **Resource Manager**

void	**SetResPurge(***doCheck* **);**	
Boolean	*doCheck* ;	TRUE=check resChanged before purging

This procedure is used by applications intending to change purgeable resources. When set, the **Memory Manager** will call the **Resource Manager** before it purges a resource handle and the **Resource Manager** will record changes to disk (if necessary).

doCheck is a Boolean value indicating whether to install or de-install resource purge checking. It is one of:

FALSE	Normal operation; no purge checking
TRUE	Before purging any handle data, check if it is a changed resource and, if it is, write its data to disk.

Returns: none

Notes: **SetResPurge** is an attempt to avoid the "silent error" of writing a 0-length resource to disk when a resource file is updated. It is needed only by applications that modify a resource tagged as resPurgeable (see **GetResAttrs**).

In a memory shortage situation the Memory Manager will discard all purgeable resources and compact memory. Then, when you **WriteResource**, **UpdateResFile**, or **CloseResFile**, a 0-length resource is sent to disk. By calling **SetResPurge(**TRUE), you can avoid this problem. Another way is to set the address of a custom "purge warning handler" into the purgeProc field of the memory manager's Zone structure obtained via **ApplicZone**. Yet another way is: avoid changing purgeable resources altogether!

Note that this does NOT keep the resource from getting purged. You must use **LoadResource** before each access of an unlocked purgeable resource. See **HNoPurge** for a way to force any purgeable handle to remain in memory.