Allocators

Among the operating-system services that Core Foundation abstracts is memory allocation. It uses allocators for this purpose.

Allocators are opaque objects that allocate and deallocate memory for you. You never have to allocate, reallocate, or deallocate memory directly for Core Foundation objects—and rarely should you. You pass allocators into functions that create objects; these functions have “Create” embedded in their names, for example, `CFStringCreateWithPascalString`. The creation functions use the allocators to allocate memory for the objects they create.

The allocator is associated with the object through its life span. If reallocation of memory is necessary, the object uses the allocator for that purpose and when the object needs to be deallocated, the allocator is used for the object’s deallocation. The allocator is also used to create any objects required by the originally created object. Some functions also let you pass in allocators for special purposes, such as deallocating the memory of temporary buffers.

Core Foundation allows you to create your own custom allocators. Core Foundation also provides a system allocator and initially sets this allocator to be the default one for the current thread. (There is one default allocator per thread.) You can set a custom allocator to be the default for a thread at any time in your code. However, the system allocator is a good general-purpose allocator that should be sufficient for almost all circumstances. Custom allocators might be necessary in special cases, such as in certain situations on Mac OS 9 or as bulk allocators when performance is an issue. Except for these rare occasions, you should neither use custom allocators or set them as the default, especially for libraries.

For more on allocators and, specifically, information on creating custom allocators, see Creating Custom Allocators.