Using Allocators in Creation Functions

Each Core Foundation opaque type has one or more *creation functions*, functions that create and return an object of that type initialized in a particular way. All creation functions take as their first parameter a reference to an allocator object (`CFAllocatorRef`). Some functions may also have allocator parameters for specialized allocation and deallocation purposes.

You have several options for the allocator-reference parameters:

- You can pass the constant `kCFAllocatorSystemDefault`; this specifies the generic system allocator (which is the initial default allocator).
- You can pass `NULL` to specify the current default allocator (which might be a custom allocator or the generic system allocator). This is the same as passing `kCFAllocatorDefault`.
- You can pass the constant `kCFAllocatorNull` which indicates an allocator that does not allocate—it is an error to attempt to use it. Some creation functions have a parameter for a special allocator used to reallocate or free a backing store; by specifying `kCFAllocatorNull` for the parameter, you prevent automatic reallocation or deallocation.
- You can get a reference to the allocator used by another Core Foundation object with the `CFGetAllocator` function and pass that reference in. This technique allows you to put related objects in a memory "zone" by using the same allocator for allocating them.
- You can pass a reference to a custom allocator (see Creating Custom Allocators).

If you are to use a custom allocator and you want to make it the default allocator, it is advisable to first get a reference to the current default allocator using the `CFAllocatorGetDefault` function and store that in a local variable. When you are finished using your custom allocator, use the `CFAllocatorSetDefault` function to reset the stored allocator as the default allocator.