Using the Allocator Context

Every allocator in Core Foundation has a *context*. A context is a structure that defines the operating environment for an object and typically consists of function pointers. The context for allocators is defined by the `CFAllocatorContext` structure. In addition to function pointers, the structure contains fields for a version number and for user-defined data

Listing 1 The CFAllocatorContext structure

```
typedef struct {
    CFIndex version;
    void * info;
    const void *(*retain)(const void *info);
    void (*release)(const void *info);
    CFStringRef (*copyDescription)(const void *info);
    void * (*allocate)(CFIndex size, CFOptionFlags hint, void *info);
    void * (*reallocate)(void *ptr, CFIndex newsize, CFOptionFlags hint, void
*info);
    void (*deallocate)(void *ptr, void *info);
    CFIndex (*preferredSize)(CFIndex size, CFOptionFlags hint, void *info);
} CFAllocatorContext;
```

The `info` field contains any specially defined data for the allocator. For example, an allocator could use the `info` field to track outstanding allocations.

Important: For the current release, do not set the value of the `version` field to anything other than 0.

If you have some user-defined data in the allocator context (the `info` field), use the `CFAllocatorGetContext` function to obtain the `CFAllocatorContext` structure for an allocator. Then evaluate or handle the data as needed. The following code provides an example of this:

Listing 2 Getting the allocator context and user-defined data

```
static int numOutstandingAllocations(CFAllocatorRef alloc) {
    CFAllocatorContext context;
    context.version = 0;
    CFAllocatorGetContext(alloc, &context);
    return (*(int *)(context.info));
}
```

Other Core Foundation functions invoke the memory-related callbacks defined in an allocator context and take or return an untyped pointer to a block of memory (`void *`):

- `CFAllocatorAllocate`, allocates a block of memory.
- `CFAllocatorReallocate` reallocates a block of memory.
- `CFAllocatorDeallocate` deallocates a block of memory.
- `CFAllocatorGetPreferredSizeForSize` gives the size of memory likely to be allocated, given a certain request.