Command-Line Tools for Analyzing Kernel Extensions

You can simplify your kext development process with the following command-line tools. More information on these tools can be found in their respective man pages.

Generate Debug Symbols and Prepare Kexts for Loading

Use the `kextutil` utility to generate debug symbols for your kext, and to test whether your kext can be loaded. While you are debugging your kext, you should use `kextutil` to load your kext instead of `kextload`.

Commonly used `kextutil` options include:

-n / -no-load
　　Does not actually load the kext into the kernel. This option is useful when you only want to generate debug symbols or determine whether a kext can be loaded.

-s / -symbols
　　Generates debug symbols for the kext in the directory specified after this option.

-t / -print-diagnostics
　　Outputs whether or not the kext appears to be loadable, along with a diagnosis if the kext doesn't seem to be loadable.

-e / -no-system-extensions and -r / -repository
　　Typically used together, these indicate that `System/Library/Extensions` should not be used as the default kext repository when resolving dependencies for your kext, and a specified folder should be used instead.

The `kextutil` utility includes additional options for simulating various load situations. See the `kextutil` man page for more information.

Output the Status of Loaded Kexts

Use the `kextstat` utility to output the following information for each kext loaded in the kernel:

- The load index of the kext (used to track linkage references)
- The number of references to the kext from other kexts
- The kernel-space memory address of the kext
- The size, in bytes, of the kext
- The amount of wired memory, in bytes, occupied by the kext
- The bundle identifier of the kext
- The bundle version of the kext

- The load indices of other kexts that the kext has a reference to

See `kextstat` for more information.

Determine Kext Dependencies

Use the `kextlibs` utility to determine which library kexts your kext must link against in order to resolve its symbols. You must list the bundle identifiers of these library kexts in the `OSBundleLibraries` dictionary of your kext's information property list.

Commonly used `kextlibs` options include:

-xml
Produces XML output you can copy and paste for the `OSBundleLibraries` dictionary of your kext's information property list.

-undef-symbols
Displays symbols that `kextlibs` could not locate. You may be able to locate these symbols by using the `kextfind` utility (see Locate Kexts).

See `kextlibs` for more information.

Locate Kexts

Use the `kextfind` utility to search for kexts with custom queries. In addition to its query predicates, `kextfind` includes predicates for generating tab-delimited reports for further processing.

Commonly used `kextfind` query predicates include:

-dsym / -defines-symbol
Prints only kexts that define the symbol specified after this option. This predicate is useful for locating symbols in your kext that `kextlibs` can't locate.

-lib / -library
Returns only library kexts that other kexts can link against.

The `kextfind` utility contains many more query predicates and report predicates you can use to fine-tune your search. See `kextfind(8)` for more information.

Obtain Instance Counts

Use the `ioclasscount` utility to obtain the current number of instances of any given subclass of the `OSObject` C++ class (which includes virtually all built-in kernel classes). The instance count returned for a class includes the number of instances of that class's *direct* subclasses. You can use `ioclasscount` to discover leaked instances that you should have deallocated before your kext was unloaded.

See `ioclasscount` for more information.

View the I/O Kit Registry

Use the IORegistryExplorer application (located in `/Developer/Applications/Utilities`) to view the current state of the I/O Kit registry. IORegistryExplorer also includes several searching and browsing options to help you navigate the registry.