Windows Internals Course – University of Tokyo – July 2003

Registry Exercises

Dragos Sambotin – 2003/07/18a

The Registry

The Registry is the centralized configuration database for the Windows NT operating system, as well as for applications. The Registry stores information about tuning parameters, kernel executive configuration, device configuration, and user preferences. Aside from Win32 APIs exposed for user mode applications to manipulate the registry, the bulk of the registry implementation lives in kernel mode, alongside with the other executive components (memory manager, object manager, scheduler, etc.) and it is known as the Configuration Manager. When accessed from kernel mode, the configuration Manager namespace maps into the object manager namespace starting at the root node '\registry'. Namespace manipulation in kernel mode is done via handle based Nt* APIs or using Ob (Object Manager) routines. For example user mode path HKEY_LOCAL_MACHINE\Software\Classes maps to '\registry\machine\Software\Classes' in kernel mode. That is the fully qualified path of a registry key object kernel mode components use.

Kernel Filter Drivers

Starting with Windows XP, a callback-like notification mechanism has been added to the operating system in order to allow for building monitoring tools of kernel mode drivers that can filter accesses to the registry. A kernel mode driver can register a callback with configuration manager and it will be notified for every operation (read/write) that is attempted against the registry namespace. Configuration manager will invoke the registered callback for each operation. For Key Create and Open the callback will be invoked twice. Once before the operation is performed (the 'pre' notification) and once after the operation has been performed (the 'post' notification). The filter driver should return STATUS_SUCCESS out of the callback, unless he wants to veto the operation. For a complete description on how the registry callback mechanism works, please consult attached "Registry Callbacks.doc" document. Additional details about type definitions and function prototypes can be found in ntddk.h

The purpose of this exercise is to show step by step how a registry monitoring tool (regmon-like) can be implemented by a kernel mode driver using the above callback mechanism. For details on how to write a kernel mode driver, please consult the DDK Design Guide under 'Kernel-Mode Driver

Architecture' section. For convenience the 'cancel ' DDK sample will be used/extended in this exercise.

Experiment 1: Modify the sample driver above to register an empty callback with the configuration manager. Upon exit, the driver should un-register its callback.

Experiment 2: Modify the driver above to count the number of values that are written to the registry during one driver uptime cycle.

Question: *If we had 'pre and post notifications, which one we would've used? (A: 'post' since we need to know if the operation succeeded – Server Only).*

Experiment 3: Keeping in mind that only one pre and one post notification is delivered to a particular thread and in this exact order, extend the above to dump in the debugger every time an attempt to create a key fails. Key name and the call elapsed time should be dumped. Hint: Store away pre notifications in a table alongside with the thread id.

Question: *What if we re interested only in keys under '\registry\machine\software\classes'*

Experiment 4: Registry is a very active component. Various OS components, services and applications read and write to the registry every time. Doing a lot of work inside the callback can lead to serious system performance degradation. A better idea is to have a client server model where the bulk of the processing work is done on a separate thread that runs in the background. Thread wakes up only when there is work to be done. Modify the driver above to dump the key name in the debugger on a separate worker thread. Hint: Queue an work item in the executive delayed work queue.

Question: *How can this be optimized further?*