

**Avoiding Memory Leaks in
WinCE Applications
Development**

By Seshu Kumar

Avoiding Memory Leaks in WinCE Applications Development

Memory leaks can be fatal to embedded and mobile applications. It's not that embedded operating systems are fragile; it's simply the way they are designed. Desktop computer systems run very fast Pentium III and IV class processors with hundreds of megabytes of random access memory (RAM), but embedded Windows CE and Windows Mobile systems run significantly slower embedded CPUs (which require much lower power) with perhaps 2 to 6 megabytes of RAM that is devoted to running 10 to 20 applications.

Memory Leak issues and types:

Because so little RAM is available in WinCE applications, mobile and embedded developers have to constantly think about how to reduce the memory footprint of their applications. With little RAM to spare, wasting the RAM by forgetting to close a handle or destroy a Graphics Device Interface (GDI) object can affect the performance of the system. Unfortunately, neither the Win32 application programming interface (API) nor the Microsoft Foundation Classes (MFC) makes it easy to avoid memory leaks. Although managed programming would solve many — but not all — of the resource problems, many developers resort to writing native code for size, performance, or real-time reasons.

The following are the types of memory leaks which occur frequently:

- Device context handles
- Heap creation and Heap allocations
- Icon Objects
- Bitmap Objects
- Leaks in GDI objects

Solution:

Microsoft has provided various remote tools to track memory leaks in Windows CE Applications.

These are:

Remote Performance Monitor tool : To monitor current memory load

This is a graphical tool for measuring the performance of a remote Windows CE system. We may view the behavior of performance objects, such as processors, memory, threads, and processes. Each performance object has an associated set of performance counters that provide information about device usage, queue lengths, and delays, and information used to measure throughput and internal congestion. Remote Performance Monitor also provides the ability to track the current activity on a target device and view data from a log file.

Remote Kernel Tracker tool: To view current memory load

Using this tool, we may view thread interactions, internal dependencies, and system state information. This tool shows all processes and threads in the system, when

they are created, run, stopped or sleeping. System interrupts and system events are also traced since they are mapped onto the thread that is executed at the time they occur.

Application Verifier Tool: To fight memory leaks

This tool assesses the stability of an application and detects common programming mistakes. The tool attaches itself to an application and performs tests while the application runs. With this tool, we may be able to diagnose subtle problems with an application that would otherwise be difficult to diagnose on Microsoft Windows CE. Application Verifier tool monitors all API calls that may allocate free handles or memory.

When the application closes, Application Verifier checks to see if all of the handles that were allocated during the life of the application were freed or released by the application before termination. Any handles or memory that the application did not free are recorded in a file, along with a stack trace, for the thread that made the allocation at the point when the handle was created or memory was allocated.

Developers do not need to do any special programming to use Application Verifier. In fact, Application Verifier can analyze applications without source code. However, if we have .map files for the application, localizing the leaks is significantly easier. Application Verifier is part of the Windows CE Test Kit (CETK) and is delivered with every version of Platform Builder - the tool used to port Windows CE to new hardware platforms.