UnForm & Windows Desktop Heap Exhaustion

Before this article can discuss Heap “Exhaustion”, it will attempt to define Desktop Heap. The below “definition” of Heap is adapted from many websites very helpful in understanding Heap, including

http://blogs.msdn.com/b/ntdebugging/archive/2007/01/04/desktop-heap-overview.aspx

http://blogs.technet.com/b/markrussinovich/archive/2010/02/24/3315174.aspx

http://www.windowsitpro.com/content1/topic/conquer-desktop-heap-problems/catpath/windows-2000

There are many other articles about Heap on line (just try 'windows heap size' in Google), including several on Microsoft's knowledge base.

What is Desktop Heap?

Windows 2000 and later systems have a configurable area of kernel mode memory called session space. Session space represents a single user’s logon environment; you can also think of session space as each user’s sandbox of windows and desktops.

Every desktop object requires memory to store UI objects, such as windows and menus, handles and objects. This memory is called desktop heap and it is not related to the total amount of memory the system has. When applications require a UI object, functions within user32.dll are called, and desktop heap memory is allocated. There is one desktop heap per desktop, and the heap memory itself is allocated from session-view space, which is a subset of session space.

Heap is configured by a registry entry:
“HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\SubSystems\Windows”

In the middle of this entry is the “SharedSection=” section that controls how desktop heap is allocated. The numeric values specified are in kilobytes.

- The first SharedSection value is the shared heap size common to all desktops. This memory isn't a desktop heap allocation, and you should not modify this value to address desktop heap problems.
- The second SharedSection value is the size of the desktop heap for each desktop associated with an interactive window station, except for the Disconnect and Winlogon desktops.
- The third SharedSection value is the size of the desktop heap for each desktop associated with a non-interactive window station (usually a **service**). If this value isn't present, the size of the

desktop heap for non-interactive window stations will be same as the size specified for interactive window stations (i.e., the second SharedSection value).

An example would be 1024,3072,512 on an earlier version of Windows like XP. As newer versions of Windows came out, one or more of these values have been increased to reduce possible problems. For example, on Win7, you may see 1024,12288,512 as the default settings and on W2008 (64-bit) you may see 1024,20480,768.

So what is Heap Exhaustion?

Simply put, Heap Exhaustion is when a process attempts to start and there is no more Heap memory to allocate to the process. So the process does not start.

On busy systems, this is not an unusual situation. If you look at the Task Manager/Processes you can use the View/Select Columns selection to display columns of data other than the default including User Objects, Handles, and GDI Objects, all of which can affect the Heap allocations. Once you have added one or more of these columns, you can click on the column header to sort and you will see what each process needs in these areas. For example, Outlook shows it uses 601 User Objects on a Win7 pc.

How does this apply to UnForm?

Like many Windows programs, UnForm is a series of programs/processes that show on the Processes tab in the Task Manager. Some of the static processes are: 1 or more uf80d.exe, uf80ss.exe, uf80svc.exe, and 3 uflist.exe. For every job that is run, there will be an uf80c.exe, ufproc.exe, and ufmain.exe. Each of these processes uses space allocated from the Heap. In addition, UnForm might utilize a support server, ghostscript, and image magick and well as processing subjobs like archiving and electronic delivery that will require heap memory.

So if you are running UnForm with a 5-job license, you could potentially have 5 copies of uf80c.exe, 5 copies of ufproc.exe, and 5 copies of ufmain.exe, each using memory allocated from the Heap.

When the Heap is exhausted, UnForm might be in the middle of starting a job where the uf80c.exe is started but the ufproc.exe and ufmain.exe were unable to start due to Heap Exhaustion. Or maybe uf80c.exe and ufproc.exe were able to start, but ufmain.exe is unable to start. The processes that started would show in the Task Manager, but would never clear because ufmain.exe was unable to process. Any processes that can’t complete will be using a license, making each license unavailable for other jobs.

Note that Heap resources come and go as processes start and end. This could explain why some processes work and others do not.

When viewing the uf80d.log file, it you see a connection line and possibly a listening line, but no Job started line, a good possibility is that the Heap is exhausted.

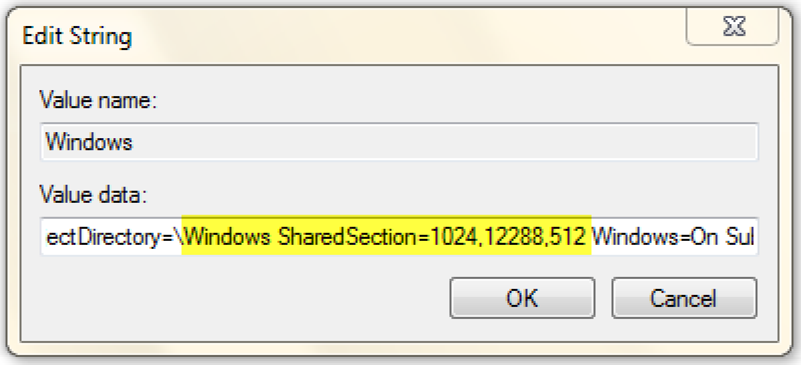
How does one resolve Heap Exhaustion?

To resolve Heap Exhaustion, you need to increase the size of the desktop heap.

According to this article on MSDN (http://support.microsoft.com/kb/126962) and modified to be more current:

1. Run Registry Editor (Regedt32.exe).
2. From the HKEY_LOCAL_MACHINE subtree, go to the following key:
   \System\CurrentControlSet\Control\Session Manager\SubSystems

3. Select the **Windows** value and double click or right mouse click and choose **Modify**.
4. Increase the SharedSection parameter.



Value name: Windows

Value data: ectDirectory=\Windows SharedSection=1024,12288,512 Windows=On Sul

For Windows NT:
SharedSection specifies the system and desktop heaps using the following format:
SharedSection=*xxxx*,*yyyy*

Add ",256" or ",512" after the *yyyy* number.

For Windows 2000, Windows XP, and Windows Server 2003:
SharedSection uses the following format to specify the system and desktop heaps:

SharedSection=*xxxx*,*yyyy*,*zzzz*

For 32-bit operating systems, increase the *yyyy* value to "12288";
Increase the *zzzz* value to "1024".
For 64-bit operating systems, increase the *yyyy* value to "20480";
Increase the *zzzz* value to "1024".

You'll probably want to increase the zzzz value to more than 1024. This is the third value and it is what gets allocated for non-interactive services. We suggest you try 2048 to start. You have to reboot after the change. You don't want to increase it too much, or the system might hit a ceiling and need to restore the registry before allowing a reboot (which is why we always recommend that a system administrator do this). For that reason, you want to inch it up until you find it corrects the problem. I've seen the allocation be as high as 12k for very large systems, but I'm not sure if they had to lower other numbers to get that working, or if there is a way to increase the ceiling.