Getting started with HeapAgent 10
if you're using
Microsoft Visual Studio

Read this first.

Contents

Setting up HeapAgent and your application

If you're developing your application with Microsoft Visual C++ for 32-bit Windows, here's how you get started with HeapAgent.

The basics

To quickly get started debugging your application with HeapAgent:

1. Compile your application with Visual C++ Program Database (PDB) debugging information and link it to the HeapAgent library (shdw32mtd.lib)
2. Use the HeapAgent Application Setup utility to configure HeapAgent for your application.
3. Run your program normally, and HeapAgent will be loaded automatically.

HeapAgent then replaces your application's heap and automatically detects a wide variety of heap errors.

Other options

HeapAgent includes several other options that you may find useful:

- When you start your application in HeapAgent, you can have HeapAgent start a Visual Studio debugger session at the same time. You can then use Visual Studio to debug your source code while you simultaneously use HeapAgent to detect and report heap errors. See "Running your application in HeapAgent," later in this booklet.
- If you want to call HeapAgent APIs from your application, or if you can't or don't want to compile with Visual C++ PDB debugging information, you can include a HeapAgent header file, **`heapagnt.h`**, and compile your application with HeapAgent debugging macros. Include **`heapagnt.h`** in each of your source files that calls HeapAgent APIs. Then define the macro constant **`MEM_DEBUG`**, as described in step 6 of the next procedure, "Setting up the Visual C++ project file."

Setting up your application for HeapAgent

To set up a Visual C++ project to work with HeapAgent:

1. Choose Open from the Visual C++ File menu, and open your application's project file.
2. Open the project's Property Pages dialog box.
3. In the Configuration drop-down list box, choose "Debug," if it isn't chosen already.
4. Click the C/C++ folder and select the General property page.
5. Under "Debug Information Format" choose Program database for edit and continue.
6. If your application doesn't call HeapAgent APIs, you can skip this step.

   Still at the C/C++ folder, in the Preprocessor property page on the Preprocessor Definitions line, add:

   ```
   MEM_DEBUG=1
   ```

7. Click the Linker folder and select the Debugging property page.
8. Set Generate Debug Info to "Yes".
9. Still in the linker folder, select the Input property page.
10. Specify the HeapAgent libraries that you want to link with. At the ***beginning*** of the

"Additional Dependencies" line, ***before*** any other object files or libraries, enter the appropriate libraries:

If you're debugging	Link with these libraries (in the order shown)
An EXE or DLL without MFC, with non-Debug MFC, or with the Debug MFC DLL	**`shdw32mtd.lib`**
An EXE or DLL with Debug MFC statically linked	**`shmfc4md32.lib`**, **`shdw32mtd.lib`**

Note The MFC integration libraries **`shmfc4md32.lib`** is in the `HeapAgent lib directory`

Choose OK to save changes to the property pages.

Configuring HeapAgent for your applications (for all Visual Studio compilers)

The HeapAgent Application Setup program lets you specify how HeapAgent and your application or applications interact, including whether HeapAgent starts automatically when you start your application and whether HeapAgent should automatically patch all of the EXEs and DLLs in your application.

You can change the configuration for an application at any time; you should do so whenever you add a DLL to your application. This ensures that HeapAgent always patches the correct set of DLLs when your application runs.

Important! HeapAgent Application Setup stores its settings in the Windows Registry under the key **`HKEY_LOCAL_MACHINE\Software\MicroQuill\HeapAgent`**. Beginning with HeapAgent 10, these settings are also stored in **`HKEY_CURRENT_USER\Software\MicroQuill\HeapAgent.`**

Note Prior to HeapAgent 10, these settings were stored **`HKEY_LOCAL_MACHINE only`**. This could caused unexpected behavior when the user did not have permission to access HKLM.

Also note HeapAgent's "patching" does not affect any of your application's files on disk. The patching occurs at runtime during process initialization, much as a debugger inserts breakpoint instructions in your application when you run under a debugger.

GUI applications

If you're developing a GUI app, you can set up an application so that HeapAgent automatically loads any time you run the application. You can run the application from anywhere: the Microsoft Visual Studio debugger, the command line, Windows Explorer, or the Windows shell.

If you don't want your application to load the HeapAgent DLL automatically, you can start HeapAgent and then run the application from within HeapAgent. For more information, see "Running your application in HeapAgent," later in this booklet.

Console applications

For console apps (apps that do not create windows) you must relink with the HeapAgent Library in order for HeapAgent to load automatically each time you run your app. You can use HeapAgent Application Setup to control:

- Whether the HeapAgent user interface is automatically launched when your application starts (if you link your app with the HeapAgent Library).

- Whether HeapAgent patches the heap routines in your application's EXE and in each individual DLL in your application.

Configuring HeapAgent

To configure HeapAgent settings for one or more applications:

1. Run HeapAgent Application Setup.
2. At Application Name, enter the full path of the EXE. If you don't remember the name or location of an EXE, choose the Browse button, and HeapAgent displays the Browse dialog box.

   Or

   To change the default settings for all applications not listed, choose **`<Global default settings>`**.

   Note Most of the settings in the Edit dialog box for the app name "**`<Global default settings>`**" affect only apps *not* listed in the Application Setup listbox. However, the Auto-Load HeapAgent DLL checkbox for **`<Global default settings>`** affects apps that *are* listed here. Unchecking Auto-Load HeapAgent DLL for **`<Global default settings>`** disables auto-loading for *all* applications.
3. Choose the Add button, and the application is added to the list.
4. Choose the Edit button, and the Edit dialog box appears.
5. Specify the settings for this application.

   Auto-Load HeapAgent DLL (GUI apps only)

   Check this checkbox if you want HeapAgent to load automatically whenever you run this application. HeapAgent will load automatically regardless of whether you run the application from the command line, from the Microsoft Visual Studio debugger, from Windows Explorer, or from the Windows shell.

   Note This checkbox is only available for GUI apps and only if you're using Microsoft Visual C++. To debug a console application with HeapAgent, you must start it from the HeapAgent user interface or link the application with the HeapAgent Library. For more information, see "Running your application in HeapAgent," later in this booklet.

   For information on *exactly* when the HeapAgent DLL is loaded, see the next topic, "When HeapAgent DLL auto-loading takes effect."

   Auto-Launch HeapAgent User Interface

   Check this checkbox if you want the HeapAgent user interface to open and minimize as an icon when your application starts. The HeapAgent user interface allows you to browse heap data, control debugging settings, and diagnose heap errors.

   If you don't check this checkbox, HeapAgent still performs full error detection. In addition, when HeapAgent detects an error, you can launch the HeapAgent user interface from the Error Report dialog box. Just choose one of the browser buttons, and the user interface launches to display the corresponding browser.

   Patch Heap Routines in this Process

   Uncheck this checkbox to prevent HeapAgent from patching any of the heap routines in your application, including the EXE and all DLLs. You should normally leave this checkbox checked. If you disable HeapAgent patching, you must link the HeapAgent Library with the EXE and with each DLL in your application. In addition, the Auto-Load HeapAgent DLL feature won't work if you disable patching, because HeapAgent can only detect errors if it replaces the heap in your application (either by patching or by being linked directly into your application).

   Important! For HeapAgent to automatically patch heap routines in your EXE or DLLs, they must either be linked with Microsoft Visual C++ PDB debugging information, or they must be linked with the DLL version of the CRT. For more information, see "Setting up HeapAgent and your application," at the beginning of this booklet.

Patch Heap Routines in Main EXE

Uncheck this checkbox to prevent HeapAgent from patching any of the heap routines in your application's EXE. HeapAgent will still patch heap routines in the DLLs you specify in the Patch Heap Routines in These DLLs list. You should normally leave this checkbox checked. One reason to disable patching in an EXE is if you're implementing a DLL and you want to perform error checking in the DLL but not in the DLL's client EXE.

Patch GlobalAlloc/LocalAlloc/HeapAlloc

Check this checkbox to enable HeapAgent patching of operating system heap APIs. By default, HeapAgent does not patch these APIs.

Name of DLL to add

If your application dynamically loads one or more DLLs with **`LoadLibrary`**, and if you do want HeapAgent to patch these DLLs, enter each DLL name in this edit line and choose the Add DLL button to add the DLL name to the listbox below. You should then use the left or right arrow buttons to move the DLL name to the appropriate list to patch or not patch the DLL.

Note By default, HeapAgent will patch all DLLs that are not present on the Do NOT Patch Heap Routines in These DLLs list *except* dynamically loaded DLLs loaded with LoadLibrary.. In most cases, therefore, there is no need to add the names of DLLs that you want HeapAgent to patch.

Patch Heap Routines in These DLLs/
Do NOT Patch Heap Routines in These DLLs

These two listboxes list all of the DLLs that your application is linked with, including the DLLs that those DLLs link with:

- The left listbox lists the DLLs that you want HeapAgent to check for memory errors. For each of these DLLs, HeapAgent will replace your memory allocator's heap routines with its own debugging routines.
- The right listbox lists the DLLs that you don't want HeapAgent to check for memory errors. By default, this listbox contains only system DLLs.

Normally, you should have HeapAgent patch all the non-system DLLs that are part of your application, including compiler class library and C runtime library DLLs. DLLs that share heap memory with your EXE or with other DLLs *must* be in the left listbox.

To move a DLL between the listboxes, highlight it and choose the appropriate arrow button.

6. Choose OK to save your changes.
7. To add more applications, repeat steps 2 through 6.
8. To save your changes, choose Save from the File menu.

When HeapAgent DLL auto-loading takes effect

Note: This section assumes that your application is not explicitly linked to the HeapAgent library shdw32mtd.lib. Historically, until the release of Visual C++ 7, this explicit linking was a seldom-used option. Now, it is the norm. If shdw32mtd.lib is not linked in, then HeapAgent will still detect errors in the application, but it will not be able to report file & line numbers for those errors; instead, you will get raw addresses

---

When you add an application name to the list in HeapAgent Application Setup, the HeapAgent DLL is automatically loaded by the operating system whenever you run the application (unless you turn auto-loading off either globally or for this particular application). However, the

operating system does not actually load the HeapAgent DLL until your application makes certain Windows API calls.

In most cases the HeapAgent DLL is loaded at the first call your application makes to any function in **`user32.dll`**. Under Windows 95/98, the HeapAgent DLL is loaded when your application creates its first top-level window. Most GUI apps make many calls to **`user32.dll`** and create their first window during initialization, but sometimes a few memory allocations will occur before this. HeapAgent handles this case just fine, but will not perform any memory checking on allocations created before the HeapAgent DLL has loaded.

If your app creates many allocations before calling **`user32.dll`** (or before creating its first window), or if your app is a console app that doesn't call **`user32.dll`** at all, you can force the HeapAgent DLL to load earlier:

- Link your application with the HeapAgent library, **`shdw32mtd.lib`**, as described under "Setting up your application for HeapAgent," earlier in this section.

  Or

- Place a call to a **`user32.dll`** function, such as **`IsWindow`**, before your first memory allocation.

Running your application in HeapAgent

Once you've compiled your application with Visual C++ PDB debugging information and linked it to the HeapAgent library, you can run the application under HeapAgent for debugging.

If you've added your application to HeapAgent Application Setup or linked it with the HeapAgent Library, no further action is required to use HeapAgent — just run your application normally from your favorite debugger, from the Windows shell, or from the command line, and HeapAgent will inform you whenever it detects an error.

Important! If your application contains DLLs, and *particularly* if it loads DLLs with **`LoadLibrary`**, please see, "Using HeapAgent with DLLs," later in this booklet.

Note You can also attach HeapAgent to a running application, or you can set up your application so it starts an instance of HeapAgent each time you start the application. For more information, see "Attaching HeapAgent to your application," later in this section.

Starting your application in HeapAgent

To start an application from within HeapAgent:

1. Choose Open from the File menu, or choose the Open button in the toolbar. The Open dialog box appears.
2. Choose the Browse button to locate the executable that you want to debug, and choose OK.
3. ***(optional)*** Specify any command-line parameters that you want HeapAgent to pass to your application, and enter the application's working directory.
4. If you haven't already configured this application in the HeapAgent Application Setup program, do so now by choosing the Setup App button.

For more information on HeapAgent Application Setup, see "Configuring HeapAgent for your application," earlier in this booklet.

5. When you're finished, choose OK.
6. ***(optional)*** Change the debugging settings, agents, and error reporting settings that you want to be in effect while your application runs.

   For more information on debugging settings and agents, see Chapter 5, "Debugging with HeapAgent," in the *HeapAgent User's Guide*. For more information on error report settings, see "Specifying where to send error reports," in Chapter 7, "Working with error reports."

8. Choose Run from the File menu, or choose the Run button in the toolbar.

   HeapAgent starts your application and (optionally) starts a Visual C++ debugger session.

Attaching HeapAgent to your application

You can debug an application with HeapAgent without starting it from the HeapAgent user interface. Just link the application with the HeapAgent Library as described in step 8 of "Setting up the Visual C++ project file," under "Setting up your application for HeapAgent," at the beginning of this booklet. When you start an application that's linked with the HeapAgent Library, HeapAgent automatically launches, attaches to the application, and minimizes as an icon each time you start your application.

If you don't want a copy of HeapAgent to be opened each time you start your application, you can prevent this by unchecking the Auto-Launch HeapAgent User Interface checkbox for the application in the Application Setup Edit dialog box. For more information, see "Configuring HeapAgent for your application," earlier in this booklet.

To attach HeapAgent to your application at any time while the application is running:

1. Choose Attach Running Program from the File menu, and the Attach Running Program dialog box appears.

This dialog box lists only programs that are currently running with HeapAgent loaded because they've been added to HeapAgent Application Setup, were started from the HeapAgent user interface, or have been linked with the HeapAgent Library, as described in "Setting up your application for HeapAgent," at the beginning of this booklet.

2. Choose the program that you want to debug, and choose OK.

Detaching a program from HeapAgent

If you're done working with a particular program but you're not done working in HeapAgent, you can detach the program from HeapAgent. The program isn't terminated, but its connection with HeapAgent is, and all data windows in HeapAgent that contain data specific to that program are closed. You can subsequently attach the current HeapAgent session to a different program.

To detach a running program from HeapAgent:

- Choose Detach Program from the File menu.

Note If you close a program that is attached to HeapAgent, it's automatically detached from that HeapAgent session. If you subsequently run your program again, it will attach to the available HeapAgent session rather than starting a new one. You must explicitly close HeapAgent when you're done with a debugging session.

Debugging multiple programs, processes, or instances simultaneously

4If the application you're debugging has multiple tasks or processes, you can debug all of these tasks or processes at once. Just run an instance of HeapAgent for each executable instance, and then start or attach to the executable as described in the preceding sections.

HeapAgent maintains debugging and error reporting settings separately for each executable that you're debugging. These settings are stored in a file that has the same name and location as your program's EXE file and has an extension of **.hpa**. For more information, see Chapter 8, "Changing and Saving Settings," in the *HeapAgent User's Guide*.

If your application is linked with the HeapAgent Library or if you've configured HeapAgent to load automatically with your application, an instance of HeapAgent starts automatically each time the application starts. If you'd like, you can prevent having separate HeapAgent sessions for each of your processes or tasks. Just uncheck the Auto-Launch HeapAgent User Interface checkbox for the application in the Application Setup Edit dialog box. For more information, see "Configuring HeapAgent for your application," earlier in this booklet.

If you turn off the HeapAgent auto-launch feature for an application, you can manually start a single HeapAgent session and then attach HeapAgent to the particular process where you want to browse heap data. At any time, you can attach the same HeapAgent session to another process.

In Win32, a DLL is mapped into the address space of the process that loaded it, and the DLL operates on the heap of that process. As a result, there's no reason to debug a DLL and its calling process separately.

Using HeapAgent with DLLs

HeapAgent normally handles all DLLs in your application automatically. However, if your application includes DLLs that share heap memory with your EXE, or if you want to debug a dynamically loaded OCX or DLL but not the EXE that loads it, please read this section.

Handling DLLs that share heap memory with your EXE

If your application includes DLLs, you may need to have HeapAgent replace heap functions in one or more of those DLLs as well as in your EXE. If any DLL allocates memory that is freed by your EXE or another DLL, or if your DLL frees memory that was allocated by your EXE or another DLL, then HeapAgent needs to replace the heap in the DLL as well as in your EXE.

You can use the HeapAgent Application Setup program to tell HeapAgent exactly in which DLLs it should replace the heap and which DLLs it should skip. For more information, see "Configuring HeapAgent for your application," earlier in this booklet.

If HeapAgent encounters an unknown DLL for which it cannot find the memory management functions and that you haven't told HeapAgent to skip, HeapAgent displays a message box when you try to run the application. The message box asks whether you want to add the DLL to the list of DLLs HeapAgent skips in subsequent runs.

If you're sure that the DLL in question does *not* share heap memory with your EXE or with other DLLs:

- Choose Yes, and HeapAgent skips the DLL. In addition, HeapAgent adds the DLL to the list of DLLs it skips automatically whenever you start an application for debugging with HeapAgent.

If the DLL in question *does* share heap memory with your EXE or with other DLLs:

- Choose Cancel, and HeapAgent stops loading the DLL.

If the DLL shares heap memory, you must relink the DLL in one of the following ways so HeapAgent can find symbols for memory allocation functions in the DLL:

- Link the DLL with Microsoft Visual C++ PDB debugging information, as described in "Setting up your application for HeapAgent," at the beginning of this booklet.

  Or

- Link the DLL with the HeapAgent Library, also described in "Setting up your application for HeapAgent."

  Or

- Link the DLL with the DLL version of the Microsoft Visual C++ runtime library:

Using HeapAgent with OCXs and dynamically loaded DLLs

If you're using HeapAgent to debug an OCX or other DLL that is loaded with **`LoadLibrary`**, you'll need to follow the procedure outlined in this section.

HeapAgent performs patching of heap routines when the HeapAgent DLL is loaded into memory. By default, it does not perform additional patching if an OCX or DLL is subsequently loaded with **`LoadLibrary`**. To enable patching of **LoadLibrary**:

1. Add the following key to the Windows NT/Win2000/XP registry:

   `HKEY_CURRENT_USER\Software\MicroQuill\`
   `HeapAgent\Apps\`<app-name>

   where <app-name> is the *full path* of your application's EXE.

2. At the registry key that you added in step 1, add an entry with the following values:
   - *Value name*: **`PatchDynamicDLLsOn`**
   - *Data type*: **`REG_DWORD`**
   - *Value*: **`1`** (one). A non-zero value means SmartHeap should LoadLibrary, and a zero value means that SmartHeap should not patch LoadLibrary.

Note: When adding the registry key (step 1 above) Use forward slashes (/) as directory delimiters. Backslash characters (\) are not legal in registry keys. For example, if your application's full path is c:\apps\foo.exe and you want to skill DLL messages on just this EXE, you'ld add the registry key: HKEY_CURRENT_USER\Software\MicroQuill\HeapAgent\Apps\c:/apps/foo.exe

In order for HeapAgent to automatically load when your DLL/OCX loads, you need to specify to HeapAgent Application Setup the name of each EXE that will load your DLL/OCX.

To use HeapAgent with a dynamically loaded OCX/DLL:

1. Run HeapAgent Application Setup.
2. At Application Name, enter the full path of the EXE that you'll use to load your OCX or DLL during debugging.

   If you don't remember the name or location of an EXE, choose the Browse button, and HeapAgent displays the Browse dialog box.
3. Choose the Add button, and the application is added to the list.
4. Choose the Edit button, and the Edit dialog box appears.
5. If you don't want HeapAgent to debug the EXE that will load your DLL or OCX, uncheck the Patch Heap Routines in Main EXE checkbox.
6. If you don't want HeapAgent to debug the EXE that will load your DLL or OCX, move any DLLs that appear in the Patch Heap Routines in These DLLs listbox to the Do NOT Patch Heap Routines in These DLLs listbox, except those DLLs that are also used by your OCX/DLL, if any.

   For example, while you're debugging a Visual C++ OCX, that OCX will normally link with the *debug* Visual C++ CRT and MFC DLLs. The application that loads your OCX (such as Visual Basic or **`regsvr32.exe`**), on the other hand, will link with *non-debug* Visual C++ CRT and MFC DLLs. You would therefore want HeapAgent to patch the debug CRT/MFC DLLs but *not* the non-debug CRT/MFC DLLs.
7. Choose OK to save your changes.

8. Repeat repeat steps 2 through 7 for each application that will load your OCX/DLL. If you're building an OCX using Visual C++ 4.0 or later, be sure to add **`regsvr32.exe`** because Visual C++ will automatically run **`regsvr32.exe`** with your OCX during the build of your OCX.

Specifying HeapAgent configuration for DLL components

You can use the HeapAgent Application Setup program to control whether and where HeapAgent patches heap routines in an individual application.

First-chance exceptions reported by your debugger

When you run your application under both HeapAgent and your source debugger at the same time, your debugger may report one or more first-chance exceptions in the HeapAgent DLL, **`shw32d.dll`**. These exceptions are caused by HeapAgent's address validation, which uses Win32 structured-exception handling. The exceptions are handled by HeapAgent and are not errors in your application or in HeapAgent.

815 6th Street South Suite 111 Kirkland, WA 98033 Phone: (425) 827-7200 Fax: (425) 650-7150 info@microquill.com